VNC

Thomas the Richter

COLLABORATORS			
	TITLE : VNC		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Thomas the Richter	December 31, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

VNC

1.1 ViNCEd Guide

ViNCEd - Guide

First Aid about ViNCEd © 1993-1997 THOR-Software inc.

Guide Version 3.21 - English ViNCEd Version 3.50

Table of Contents:

Thomas Richter

Rühmkorffstraße 10 A

12209 Berlin

Germany

EMAIL: thor@einstein.math.tu-berlin.de

WWW: http://www.math.tu-berlin.de/~thor/thor/index.html

1.2 The THOR-Software Licence

The THOR-Software Licence

This License applies to the computer programs known as "ViNCEd" and "SetVNC", and the "VNC.guide". The "Program", below, refers to such program.

The programs and files in this distribution are freely distributable under the restrictions stated below, but are also Copyright (c) Thomas Richter. Distribution of the Program by a commercial organization without written permission from the author to any third party is prohibited if any payment is made in connection with such distribution, whether directly (as in payment for a copy of the Program) or indirectly (as in payment for some service related to the Program, or payment for some product or service that includes a copy of the Program "without charge"; these are only examples, and not an exhaustive enumeration of prohibited activities). However, the following methods of distribution involving payment shall not in and of themselves be a violation of this restriction:

(i) Posting the Program on a public access information storage and retrieval service for which a fee is received for retrieving information (such as an on-line service), provided that the fee is not content-dependent (i.e., the fee would be the same for retrieving the same volume of information consisting of random data).

(ii) Distributing the Program on a CD-ROM, provided that the files containing the Program are reproduced entirely and verbatim on such CD-ROM, and provided further that all information on such CD-ROM be redistributable for non-commercial purposes without charge.

Everything in this distribution must be kept together, in original and unmodified form.

Limitations.

THE PROGRAM IS PROVIDED TO YOU "AS IS," WITHOUT WARRANTY. THERE IS NO WARRANTY FOR THE PROGRAM, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IF YOU DO NOT ACCEPT THIS LICENCE, YOU MUST DELETE ALL FILES CONTAINED IN THIS ARCHIVE.

1.3 Overview over ViNCEd

ViNCEd © 1996 THOR-Software - Version 3.32

Written by Thomas "the Judge" Richter.

ViNCEd is a full-screen windowhandler replacement for the console device and the related DOS drivers, CON: and RAW:. The whole window content is kept in memory, so you may use the cursor keys to move upwards on the screen and re-execute old commands by pressing the RETURN key over them, like in the 8-Bit times. See also Remarks about the line buffer .

Like with the Commodore handlers CON: and RAW:, ViNCEd supplies the clipboard: You may mark text with the mouse, copy and cut them and may re-insert them later into a ViNCEd window, or any other program.

Like the XTerm program under UNIX, ViNCEd windows come with two scrollers to reach the lines scrolled off the window. Additionally, you may define keyboard macros and so called buttons, gadgets in the window title that invoke macros, too.

Macros come with parameters that are filled thru the user on invocation time by calling a file requester.

Special control characters in the window title can be used to display information like the path or the return codes of the shell running in this window, see here .

The menu controls some of the settings of the window, macros can be defined by them, and other useful stuff, like clipboard control, is found here.

Of course, ViNCEd supports a history to keep your old commands, additional to those commands reachable thru the window. As a special feature, ViNCEd offers you a special key, Control-Z . Like in UNIX machines, this key sends the running process into the background and forks a new shell.

ViNCEd can be controlled thru ESC-sequences , which are a superset of the console.device's commands. Some useful commands of the UNIX XTerm and the SGI WinTerm have been added, and a lot of private commands complete the set.

A standard ANSI color layout is available, as well as a VT-220 emulation .

The settings of this editor can be controlled thru a single program, called SetVNC , which can be called either from workbench as a preference editor or from shell to change the settings on line.

Last but not least, the vnc.library supports not only window control functions, but also support for writing an own editor.

OOps, just forgot the last point... It's free.

But please read the licence agreement .

For flames or bug reports, consult the author .

1.4 The keyboard

A short guide to the ViNCEd keyboard layout

Cursor keys : Move cursors thru the window, scroll at window borders. If you selected the "Alternate Movement" menu item , the cursor up and down keys move in the history and not in the window.

Shift + cursor : Move to the top, bottom or to the start and end of a line. With "Alternate Movement" enabled, Shift cursor up/down search in the history .

Ctrl + cursor : Scroll to the top, bottom, left or right end of the line or the window.

Alt + left/right: Move cursor to the next or preceding word.

Alt + up/down : Move in the history . With "Alternate Movement", this will move the cursor in the window.

Shift+Alt up/down: Search in the history . With "Alternate Movement", this moves the cursor to the top or bottom of the window.

The next four sequences work only in the shell mode since they are supposed to operate on path names:

Ctrl+Alt left : Move to the beginning of the preceding path component, i.e. the previous group of characters separated by a slash or a colon.

Ctrl+Alt right : Move to the next path component.

Ctrl+Alt Del : Delete the path component under the cursor.

Ctrl+Alt BS : Delete the preceding path component.

Some other delete style operations follow:

Shift+Del : Remove the complete line.

Alt+Del : Remove only user inputs from the current line.

Alt+Backspace : Delete the word under the cursor, i.e. the next group of characters separated by spaces.

Shift+Alt+Del : Delete from the cursor position to the end of the line.

Shift+Alt+BS : Delete from the line start to the cursor position.

REMARK: The meaning of the four keyboard combinations above can be exchanged on request. The flag is located at the fifth editor page of SetVNC.

TAB : Insert spaces to the next TAB stop, or jump to it, if the "TAB-Move Only" menu-item is checked. If the TAB expansion is enabled and your window is a shell window, then the file path under the cursor is expanded, fully or partial, depending on the settings, or the next possible expansion is inserted.

Shift+Tab : Removes characters up to the previous TAB stop, or jumps backwards to it. If the TAB expansion is enabled in a shell window , it also expands a file path, but it moves backwards in the list of possible expansions. If the line is an empty line, the Tab expansion will result in the list of available devices.

Ctrl+Tab : This works like TAB if the TAB expansion is disabled and the window is a shell window ; hence it expands the the file name under the cursor. If the TAB expansion is enabled, this works like the TAB key: It inserts spaces or moves the cursor.

Shift+Ctrl+Tab : Works like Shift+TAB if TAB expansion enabled, see above.

Esc : The next control code is directly inserted - like with the build-in screen editor of the ATARI XL series. This MAY confuse the shell - if you insert a return character. The inserted control code is displayed in reverse video. For example, the Esc sequence itself is inserted by pressing the Esc key twice, and is displayed like an inverse "[". This is very useful for putting Esc-sequences into the text. Depending on the keymap, a CSI control character can be obtained by "Esc Alt+Esc"

If the numeric key pad mode is enabled, the following numeric keys can be used for cursor control, too:

Numeric 8 & 2 : Replaces ALT+Cursor up and down. Hence moves in the history, or moves the cursor with Alternate Movement turned on.

Numeric 4 & 6 : Moves the cursor leftwards or rightwards, same as the plain cursor left and right keys.

Numeric 9 & 3 : Moves the cursor one page upwards or downwards, like Shift+cursor Up and Down.

Numeric 7 : Moves the cursor to the "Home" position, i.e. to the leftmost topmost position.

Numeric 1 : Moves the cursor to the "End" position behind the end of the bottommost line in the window.

Numeric 0 : Toggles between insert and overwrite mode.

Numeric 5 : Moves the cursor to the end of the line.

The key Alt+NumL (the key above the "7" on the numeric key pad) can be used to toggle the keypad mode manually. (New in 3.41)

Some mouse based operations follow:

Left Mouse : Move the cursor to the marked position, or mark a block by "dragging" it. If you press the left mouse button together with the Ctrl-key, this works like the middle mouse button. The mouse positioning is disabled if you enabled the XTerm mode .

Middle Mouse : Well, I know not everybody has a mouse with three buttons - but as you might have noticed, I have. The old versions of the operating systems - before 1.3 - do not support it at all. For this reasons, the replacement Ctrl-Left Mouse is needed.

However: The middle mouse button moves the cursor to the position the mouse points to and inserts the text in the clipboard to the current position. This works somehow like in the GNU editor EMACS and like the clipboard support of XWindows under UNIX.

Starting with release 3.30, you may disable the middle mouse button and use it for other purposes, see the docs about the prefs editor how to do this. The replacement sequence Ctrl-Left is, however, still valid.

Help : Well, calls the help macro to get this guide, unless you changed the help macro .

Ctrl-\ : Sends either an EOF to the reading process, or one of the two window close macros, depending on the "Smart Close" menu selector. The EOF-feature works only in the newer version of the OS, since the old 1.3 shell-seg is a bit picky about it (it trashes the memory when the input stream is shut down).

Starting with release 3.19, the "Smart Close" got even smarter: It looks if a program is running in the shell and uses one of two possible close macros: The "Quit shell" macro if the shell is waiting for a command - or the "Quit program" macro if a program is currently running. If the selected macro is empty, an EOF will be used instead.

Both macros can be setup by the SetVNC program , either here or here .

The function keys F1 to F10 are are assigned to the Buttons in the title bar, from the left to the right - so the keys on the keyboard are ordered in the same direction as the gadgets in the title. The ordering has changed in 3.32, sorry for the mess...

Ctrl-C to -E : Send break signals to the active shell. The way how a break is send differs a bit from the usual CON: handler. With Ctrl-C, ViNCEd will even break "Background Processes" - processes started with the "RUN" command. This is not totally compatible to what "CON:" does, but I think the better choice in an emergency situation.

But if you insist, press...

Shift+Ctrl-C..-E: Sends the break signals in a way compatible to "CON:". Does not break "Background Processes".

REMARK: This notation of "Background" has nothing to do with the ViNCEd job control "foreground" and "background" processes. In fact, an Amiga shell "Background Process" is still, for ViNCEd, in "foreground", and may trash your output (confused ?). If you want a program to execute in background, better use the "fork" script .

To find out more about job control, read here .

The next ones are left over from CON:

Ctrl-A : Like in XTerm with tcsh or bash, this moves the cursor to the start of the line.

Ctrl-K : Kill the part of the line behind the cursor.

Ctrl-Y : Yank (re-insert) the part of the line cut before with Ctrl-K.

Ctrl-S : Stop output, like with VT-100 terminals.

Ctrl-Q : Enable output. If ViNCEd seems to be "locked", try this one first to unlock it.

The remaining keys are very like the usual keys you get with CON: (for example, pressing an "A" gives "A", got it ?:-)

1.5 ViNCEd menus

A standard ViNCEd window has four menus: the Project and Edit menu, and the Macros and Settings Menu.

The project menu has functions for opening, closing and other stuff like this.

New Window : Opens a new ViNCEd window by calling the New Menu Macro .

The next four menu items are new to 3.50. They provide functions that used to be implemented in SetVNC in previous releases, namely saving and loading of various ViNCEd buffers:

Open... : Reads the window contents from an ASCII file. A previously saved ViNCEd session saved with "Save As..." can be restored like this. Similar to the SetVNC GET SCREEN function.

Open History... : Loads the history buffer from an ASCII file. Similar to the SetVNC GET HISTORY command.

Save As... : Saves the window contents, including character styles, to an ASCII file. All rendering informations are saved in ANSI/ViNCEd compatible CSI sequences. Similar to the SetVNC PUT SCREEN command.

Save History... : Saves the history buffer in an ASCII file. Similar to the SetVNC PUT HISTORY command.

Expand Window : Resizes the window to the full text overscan size and moves it to the foreground.

Shrink Window : Shrinks the window to the minimal possible size and moves it to the background.

Next Screen to Front: Moves the top most screen to the back. This is useful under KickStart 1.3, where Left-Amiga-M is just crap.

Iconify : Iconify this window - works in the same way as the iconification gadget in the window title. If the iconification is not allowed, this item gets disabled.

Jump to Next Screen: The ViNCEd window jumps to the next public screen available. If the window can't be closed for some reason or ViNCEd was opened on its own screen, this item gets disabled.

Help : Does the same like pressing the HELP-key. This, as the next action, will call the help macro in a shell window .

About : Just a tiny submenu which presents the copyright and the version (which should be 3.50 or better).

Close Window : Closes the window thru the selected close window macro or sends an EOF, depending whether or not the "Smart Close" menu item is selected.

The edit menu controls the block operations of ViNCEd:

Cut,Copy,Paste : The usual block stuff...

Hide : Unmarks the marked block.

Select All : Selects the complete contents of the window as a block. This might be quite a lot!

To put in other words: Use at your own risk.

Clear Window : Clears the complete lower window buffer , that is the window itself and all lines below. In a shell window - as a hack - this also simulates a RETURN key press to re-display the shell command prompt.

Erase to End Of Window: Clears everything behind the cursor and below the current line.

Reset Terminal : Resets the terminal settings to defaults. In special, all settings that can be controlled by ESC or CSI sequences are reset to the global preferences, thus some of your private favorites for this window might get lost. Same as the control sequence "ESC c".

As a new feature, this menu item will also erase the upper line list and the history if selected with Shift hold down.

Search Forward : Search forwards in the history like Ctrl+Alt+cursor up. See also: The keyboard

Search Backward : Search backwards in the history.

The macros menu invokes the macros , but also gives you the ability to define them thru the submenu. Mark the supposed macro text by "dragging" it (create a block) and select the proper item of the "Cut Macro" submenu you want to replace. Special keyboard presses can be either inserted by backslash sequences, or by intelligent use of the ESC key . A line break is translated to a "Return"-key press.

The backslash sequences for the macros can be found here .

The Settings Menu

Some of the settings of ViNCEd can be controlled thru the settings menu; the complete set, however, can be accessed by SetVNC . (Probably more than you ever need...)

Cut History Size : Cut a number, marked as block, and interpret it as the number of lines in the command history .

Cut Display Size : The number of lower lines of the display buffer is being cut out.

Cut Upper Size : Same as above, but for the upper lines, i.e. lines scrolled above the top edge.

And now *some* settings:

Dos Cursor Mode : Same as "Unrestricted cursor movement" in the SetVNC edit menu, more can be found here .

Overwrite Mode : "Overwrite text", also here .

Wrap Around Buffer : "History buffer wraps around", again here .

Smart Close : "Call macro to close window", for more about this, consult again the SetVNC part about the Shell menu.

Safer Close : "Prevent accidental window closing", again in SetVNC . This item must be checked, too, to get the close requester. See here .

Cut Inputs Only : "Don't write printed text into clipboard". This can be found here .

Rebuild Delay : "Allow delayed window refresh" This is part of the timing menu in SetVNC .

Auto Copy : "Implicit copy after text marking" Again, this can be found in the edit menu of SetVNC.

Alternate Movement : "Swap ALT-cursors and cursor keys", can be found here .

Expand with TAB : "TAB filename expansion". See in the shell menu description of the SetVNC program .

Double TAB Requester : Same point in the SetVNC menu: "Double TAB Requester", see there .

Request if not Unique : Same as "Requester if expansion is ambiguous", again here .

Expand fully : "First TAB expands fully". Also, here .

Partial Path : "Search commands in current and C: only". Another TAB expansion setting, used to speed up command searching and useful for people with very long paths. Read on here and also here for another flag interacting with this one.

The later options control all the behaviour of the TAB expansion, read on here about this feature!

1.6 ViNCEd Macros and Buttons

The macros can be controlled thru the macros menu of a ViNCEd window, or by the menu page of the SetVNC program, read

here for macros or here for buttons to find out how to control them by SetVNC.

What is a macro, conceptionally?

A macro is a text, together with some special control functions, which is send to the keyboard buffer once executed. Thus expanded macros are kept like typed user input in the ViNCEd window buffer . Macros are invoked by the ViNCEd Macros menu or the "Amiga-Key" shortcuts. Some macros are kept for internal use.

And what about buttons?

A button is a gadget, which is put into the title bar of the window. It offers the same functionality as a macro and uses the same syntax for expansion, but has an additional title, or so called "short cut". This short cut is the text that is put into the title bar (since the expansion text might get long).

How to define them?

You can define a macro by typing its contents into the window, mark it as a block and select the proper submenu of the cut macro menu-item . Your new macro replaces then the old one present there at selection time.

Unlike macros, buttons can not be defined so easily. You must use the SetVNC program to change or add the titles or contents of the buttons. However, they are in principle not limited in number, except for available room in the window title bar and for available memory. Read here to find out how to change the buttons once you called the SetVNC program.

What about special characters like RETURN I want to add to macros?

Now to the special characters that can be used in the definition of a macro (not in the definition of a button title, but in its contents):

To include preceding and following blank spaces in a macro, enclose the whole macro string into double quotes (\").

There are, however, more special characters that can be used in macro strings as well:

\" : The double quotes enclose white spaces

' : Same goes for single quotes

\ : The backslash escapes the next character, like:

\n : The Newline code, works like Ctrl-j and is often not very useful, with one exception (see below).

\r : The CR code, which is send by the RETURN-key.

\t : The TAB key

\b : The Backspace key

\f : The form feed code, works like Ctrl-l and erases the lower part of the ViNCED display buffer .

\\ : The backslash itself

\\" : The double quotes

\' : Single quotes

\code. : The given decimal code is inserted. E.g. \"\65.\" inserts the letter \"A\". Don't forget to put a dot at the end of the code. This dot MAY be omitted if no digit follows.

\$code. : Inserts an hexadecimal value.

A special code can also be inserted into a macro by typing it directly to the window, using the ESC-key .

It is then converted to the form presented above. To give an example:

Press ESC backspace to insert a backspace sequence to the window text, without executing it, i.e. deleting the character in front of the cursor. Cutting the resulting inverse "H" on the screen as part of the macro will insert the backspace code in the macro body.

[ ] : The square brackets group patterns. The pattern in between is taken as a file wildcard, and a requester is build. After the user selected a file, the path is inserted. If the user cancels, the macro expansion is aborted. To allow file names with spaces, it's always a good idea to give an extra pair of double quotes, i.e. use \"[..]\" instead.

[@] : As a special pattern, this expands into a unique ID as a decimal number. (Hopefully nobody will call a file @...)

How to see macros and buttons in action?

The macros are invoked by the Macros Menu , and, as a shortcut, by Amiga-0 to Amiga-9.

A similar shortcut exists for the buttons, which are usually activated by a press to the buttons in the window title bar. The shortcut for the leftmost button is the F1 key , the next button to the right has the shortcut F2 key assigned to it and so on.

Except the ten user definable macros, ViNCEd has an additional set of six system macros. They cannot be defined in the window, but with the SetVNC program.

Here their meanings:

Quit Shell : This macro is called whenever a shell running in the window must be closed and the "Smart-Close" menu item in the settings menu is checked. It is supposed to be used by 1.3 owners, cause the usual 2.0 close by sending an End Of File to the shell would hang the system. Also, it is advised that users of the newer OS releases use this way to close the window, cause vital cleanup procedures could be run by this system macro (i.e. remove all temporary files when leaving a compiler session). If this macro is left empty, an EOF will be sent to the shell.

However, ViNCEd does not matter how it closes its window.

Starting with 3.19, there's a second macro that can be invoked to close the window. It is called "Quit program" and is used if a module loaded by the shell has to be aborted.

New Window : Called, if the user selects the "New Window" item of the project menu. Should fork a new shell in a new ViNCEd window. Insert here your favorite shell command, and as a window parameter your favorite shell editor, which should be ViNCEd... :-)

Get Help : Answers the user's call for help. Invoked by the "Help" item of the project menu and by the help button (if not in RAW-mode).

This macro should present some useful help, e.g. should call MultiView with an application specific online guide. As default, this guide is presented.

Again, an example would be to display the compiler's guide in a compiler session.

Edit Settings : Called, if the user selects the "Settings" item of the project menu. This should call a specific settings editor, or, as default, the ViNCED SetVNC .

Fork new Shell : This one is unique, cause it does not run into the keyboard buffer, but is executed directly by AmigaDOS. It is called, when the user presses Ctrl-Z to get a new shell prompt, i.e. to send a running process into background. Cause this is NOT parsed as keyboard presses, the command should not be terminated by a RETURN-press \"\r\", but by a line feed \"\n\".

Quit program : The second macro that is used to close a ViNCEd window, if the "Smart Close" option in the settings menu is checked.

Unlike the "Quit" shell macro which is used if a plain shell running in the window must be aborted, this second macro will be called if the shell is executing a command. It should shutdown the program and/or close the window, probably by sending a Ctrl-C .

1.7 Path Arguments

A lot of arguments can be given in the open path; most of them are compatible to Commodore's CON:-arguments, some are compatible to the old ConMan, and some are new to ViNCEd:

VNC :[W userwindow/][S userscreen/][B superbitmap/]

leftedge/topedge/width/height/title[/WAIT][/AUTO]

[/CLOSE][/NOCLOSE][/SMART][/SIMPLE][/INACTIVE][/BACKDROP]

[/NOBORDER][/NOSIZE][/NODRAG][/NODEPTH][/WINDOW window][/KEEP]

[/SCREEN pubname][/ALT leftedge/topedge/width/height][/MENU]

[/NOMENU][/NOPROPX][/NOPROPY][/PROPX][/PROPY]

[/FALLBACK][/OLDLOOK][/STITLE screentitle][/CHUNKY][/PLANAR]

[/SHELL][/NOSHELL][/PREFS pathname]

[/BUTTONS][/NOBUTTONS][/ICONIFY][/NOICONIFY][/ANSI][/NOANSI]

[/SDEPTH planes][/MONITOR name][/MONITORID id][/FONT name]

[/COLS width][/ROWS height][/TITLEBAR][/NOTITLEBAR][/windowflags]

Everything in square brackets is optional. (That's quite a big bunch of optional.... uhm... options. :-)

If you need slashes (/) in a string argument, you should enclose this string in double quotes, or escape it with a backslash (\). It is also used to produce the double quotes and the backslash itself:

\" : Double quotes

\\ : The backslash itself

\/ : The ordinary slash

For example, the string

"This is / a \"test\" \\."

produces the title

This is a / "Test" \.

Remark, however, that the shell itself parses an argument, as given to the NewShell command, again; so if you want to use this string as an argument to the shell, all double quotes should be escaped with the BCPL escape-character * itself. The following command will result in a window with the desired title presented above:

NEWSHELL "VNC:////*"This is / a *"test*" \\.*""

All empty arguments are filled with default values, here the window position. (Rather tricky, ey ?)

Remark: The old versions of AmigaGuide do not parse the backslashes correctly. If the example string above ends with four backslashes instead of two, you use an old version of this program and you should update your AmigaGuide browser.

And now the option's part (Supposingly, optional.)

Here the description of the arguments:

W userwindow : Put a ViNCEd into an already opened window whose address is given as a hexadecimal string "userwindow". You may want to add an explicit "$" or "0x" to mark that this is a hex number, although this is not strictly necessary. This argument is compatible to the old ConMan.

REMARK: If you give the KEEP option as well, ViNCEd does not close the window if the channel is closed. This behaviour has changed in 3.30 (the KEEP option was implicit in earlier releases).

S userscreen : Open a ViNCEd window on a previously opened user screen at the given address, which is in hex again. Also compatible with ConMan. (Note that the "userscreen" is an address, not a name of a public screen)

B superbitmap : "superbitmap" is the address of a bitmap structure which is to be used by the ViNCEd-window. However, the use of superbitmaps is not recommended.

leftedge/topedge/width/height/title : Standard arguments. You may want to omit them, then defaults are chosen. Also, negative values for the width and height are fine: In this case, the dimensions of the text overscan range of the screen is used. The value of "-1" for the topedge coordinate is fine as well: It will adjust the window precisely to the bottom line of the screen drag bar.

REMARK: AT LEAST one of the arguments MUST be given. So opening a window with no description at all, like with "VNC:" would produce an error. The minimal possible path is "VNC:/". This is again a hack that makes it possible to pre-mount ViNCEd, e.g. to have "Activate = 1" in the MountList entry. I am sorry about this hack, but I couldn't find out how else a handler is informed if it is just pre-mounted instead of being run.

ANOTHER REMARK: Starting with release 3.30, C-style control sequences can be embedded into the window title. They get expanded into information about a shell running in this ViNCEd window. Read on here to find out more about them.

WAIT : After all DOS-processes closed their streams to the window, it won't close, but remains open until the user closes it explicitly. This is compatible to the same CON: - option.

AUTO : Does not open the window immediately, but waits until a "Read" or "Write" is requested for this window. Also, if there is no read or write pending, the user can close the window. It later pops up again, when a DOS request is send to it.

The specific behaviour of this argument is controlled by the mystic "Auto Close Queue" menu item in the settings menu, or by the "Don't send EOF until everybody waits" option in the shell menu of the SetVNC program. If checked, a window close message (either EOF or invocation of the close macro) is send only if all streams of the window wait for input. Otherwise, the window just closes. If not checked (the default) the EOF is send if one process waits for input. Since this is rather technical, I would advise you to leave this item in its default position.

This option is also compatible to the CON: handler.

REMARK: There is another quirk with this option. There is a documented DOS-packed that reads the window pointer. Once this pointer is made public, ViNCEd can't close the window again since somebody could still use this window. So after starting a program that reads this pointer, the AUTO-option gets inactive. An example of such a program is the utility "more", so I "recommand" you to use "Multiview" instead. A second problem that is caused by "more" is, for the same reason, that the window can no longer be iconified. A script , also called "more" has been put into your S: drawer at installation time for a work-around. It calls first "more" and frees the window pointer afterwards with the SetVNC program explicitly.

ANOTHER REMARK: (for program authors only) If you DEFINITELY NEED the intuition window pointer, use the vnc.library function FindCNWindow(stream) which gives you a pointer to the CN-window main structure. The window-pointer can be read out of this structure. Once you are done with this pointer, call UnFindCNWindow(stream) to re-enable the AUTO-feature and to give the pointer back. To find out how this structure looks like, contact me or read the docs.

CLOSE : Add a close gadget to the window. This is done by default since the very first ViNCEd version (version 1.00 of VNC), but to be fully compatible with CON:, it can be disabled thru the "Don't add close gadget by default" check mark in the SetVNC shell menu .

NOCLOSE : Do not add a close gadget. Since usually all ViNCEd windows come with one, you should add this option if you do not want a close gadget.

SMART : Use a smart refresh window instead of a simple refresh window. They are a bit faster on refresh if a hidden part of the window is brought to front, but use more chip memory. Since ViNCEd is written in 100% assembler, the difference is not very big an can be seen with the antique MC68000 processor on board only. I advice you not to use this option.

SIMPLE : Use a simple refresh window. This is also the default and the recommended way.

INACTIVE : Do not activate the window when opening.

BACKDROP : Open a backdrop window. This is seldom useful, except to use a complete screen as ViNCEd output. You should use the noborder-flag of the window flags described below, no title, no sizing and drag gadget, no scrollers, no buttons and no close gadget to produce a nice invisible window. If you fail to do so, the result may look rather ugly.

NOBORDER : Don't put a border around a window. Only useful if you do NOT give a title, too. This might be rather confusing to the user, so use with care!

NOSIZE : Do not put a size gadget into the window.

NODRAG : Don't allow the user to move the window. I have no idea why somebody should use this flag; however, it is here to be compatible to CON:

NODEPTH : Don't allow to re-arrange this window. The zoom gadget is also missing if you give this option, but this is a "feature" by intuition, not by ViNCEd. Again, this option is rather useless and user-unfriendly.

WINDOW window : Put the ViNCEd window into an already opened "intuition-window". This does the same as the ConMan-compatible "W"-option, like there the number defaults to hexadecimal. It always did, only the guide was wrong... sigh!.

WARNING: This is for wizards only!

KEEP : Makes only sense if the WINDOW argument or the ConMan W option is given as well. This prevents the custom window from being closed if the stream shuts down, as it has used to be in the 3.20 release. I still think the behaviour without KEEP is somewhat strange, since ViNCEd has to close a system resource it hasn't opened. This might cause trouble if you added custom gadgets or other nice stuff to the window -specify KEEP to remove them safely and close the window by hand later.

ViNCEd is also a bit "friendlier" to "KEEP" custom windows. For example, it does not try to resize them.

SCREEN pubname : Unlike the ConMan-S option, open ViNCEd on a public screen with given name. If the screen is not there, fail to open. If the FALLBACK option is given too, it opens on the workbench in this situation.

WARNING: If the name of the screen contains blank spaces, enclose it in double quotes. An additional asterisk "*" is needed in front of these double quotes if you give the path as an argument to a shell command (due to the parsing done by the dos library).

ALT left/top/width/heigh: Specify the alternate window position which is used on a press of the zoom-gadget.

MENU : Add the default menu to the window - this is the default.

NOMENU : Do not add the default menu to the window. Currently, there is no way to disable the menu by default, so add this option if you do not want it.

NOPROPX : Do not add the horizontal scroller .

PROPX : Since the scrollers can be disabled by default , this option can be used to turn the horizontal scroller on explicitly.

NOPROPY : Do not add the vertical scroller .

PROPY : Add it explicitly.

FALLBACK : If the public screen given with the SCREEN-option does not exist, fall back to the default public screen, usually the workbench.

OLDLOOK : Show the menus in the old, pre 3.0, look. This has an effect only if running under kickstart 3.0 or better.

STITLE screentitle: Define the screen title that is shown when the ViNCEd window is the active window. Defaults to the screens default title.

REMARK: Special characters are available to setup a dynamic screen title. See also the paragraph about the window title.

CHUNKY : Do not use the bitplane optimizations for Amiga bitplanes. This speeds up ViNCEd if run with a graphics card (at least I hope so...)

This option can be turned on by default thru the "Chunky pixel graphics" switch in the SetVNC System menu .

PLANAR : Counterpart of "CHUNKY": Use optimizations to speed up scrolling on Amiga screens.

SHELL : Define this window to be a shell window. This enables the TAB-expansion thru Ctrl-TAB or TAB and the Ctrl-Z job control. Also, enables the buttons and the menu. This option can be turned on by default in the SetVNC Shell menu .

NOSHELL : Disable job control , TAB expansion , buttons and menus , counterpart of "SHELL".

PREFS pathname : Load the preferences from the specified path, and use them in the new window. Way more settings can be influenced in this way than thru the options presented here.

The preferences should be edited and saved with the SetVNC program .

BUTTONS : Add the user defined buttons to the window. This is also the default.

NOBUTTONS : Do not add the buttons .

ICONIFY : Add an iconify gadget to the window. This is done by default if the screen is the workbench screen, otherwise this gadget is omitted. Be warned: Iconification is DISABLED if some program reads the window pointer, like the "more" pager. Once disabled, the iconify gadget just shrinks the window and puts it to background.

NOICONIFY : Do not add an iconification gadget.

REMARK: The design of the iconify gadget was adapted by the 4DWM-window manager iconification gadget, the window manager that comes with the SGI computers.

ANSI : Enable the ANSI coloring scheme . In this mode, ViNCEd interprets the arguments of the set rendering control sequence CSI m not as pen numbers, but as ANSI colors. ViNCEd tries its best to display a color similar to the ANSI standard.

This mode is also selectable from the Prefs Editor , or by an CSI sequence .

NOANSI : Disable the ANSI coloring explicitly. This is also the default.

FONT name : Specify a default font for the window. This argument is given as "name.size", where "name" is the name of the font without ".font", e.g. "topaz.8". This option overrides the default font setting of the SetVNC program. See here .

COLS width : Sets the width of the window, in units of characters of the used font.

ROWS height : Sets the height of the window in characters.

If one of the next three arguments is present, ViNCEd will open its window on a private screen. If you specify the SCREEN argument as well, a public screen of this name will be created if not already open. The width and height of this screen will be taken from the width and height of the window. Remember that you might specify "-1" for these values to open a window, and hence a screen, at the maximal (text overscan) dimensions:

SDEPTH : This specifies the depth of the screen ViNCEd should appear on. If this argument is given, a private screen will be created. The argument is the number of bitplanes, and given in decimal. This arguments defaults to 2 (=4 colors).

MONITOR : This specifies the name of the "Monitor" ViNCEd will use to open a private screen.

REMARK: The monitor names can be seen in the "ScreenMode" prefs editor. But be warned: THE MONITOR NAMES will be LOCALIZED, so a ViNCEd path using the MONITOR argument working in one language might fail to work in a different language!

MONITORID : This is also used to specify the monitor for a private screen. However, this time the hex ID of a monitor is used. The default is here 0x8000: Use the default monitor, in HIRES. The default of the monitor options can be selected by SetVNC .

TITLEBAR : Display a title bar in the custom screen. This is the default.

NOTITLEBAR : Suppress the titlebar. This makes only sense if specified together with the BACKDROP option. In this case, the title bar will be moved behind the ViNCEd backdrop window.

windowflags : A ConMan compatible stream of cryptic letters, also used for control of the window appearance. For example, the ARexx tool "TCO" uses them. Strange, but such is life!

The possible flags are sometimes identical to the options given above, but they are all toggle on/off flags, i.e. giving one twice is as good as giving it not at all.

Z (GimmeZeroZero) : Produce a GimmeZeroZero window. Much slower in workbench 1.3 and below, and rather useless.

B (Backdrop) : Like the BACKDROP option.

N (Borderless) : Like NOBORDER

R (SimpleRefresh) : Like SIMPLE, the default.

M (WindowDrag) : Like NODRAG.

D (WindowDepth) : Like NODEPTH.

S (WindowSizing) : Like NOSIZE.

C (WindowClose) : Like NOCLOSE.

A (Activate) : Like INACTIVE.

The next ones are new to ViNCEd and not present in ConMan:

L (Luxury) : Use a simple version version of the window, with no scrollers, no menu and no buttons. The DOS cursor movement is activated as well. This is left over from VeryNewCon 1.xx and was never used... sigh.

O (AddMenu) : Like NOMENU.

F (PubFallBack) : Like FALLBACK.

X (AddPropX) : Like NOPROPX.

Y (AddPropY) : Like NOPROPY.

G (AddButtons) : Like NOBUTTONS.

1.8 ANSI coloring scheme

If you send a "set render" CSI sequence to ViNCEd, the argument supplied to this sequence is usually read as a pen number of the screen ViNCEd displays its window on. This is due to compatibility to the CON: window handler.

However, you may select the ANSI coloring instead, where the arguments refer to special colors, not to pen numbers. Additionally, the standard color layout is chosen - to match the ANSI specifications, namely white text on black background.

The ANSI coloring is available either as default , or as an option in the open path , or can be selected by an CSI sequence , online.

The following table gives an overview about the ANSI color definitions and how they are activated. This color list has been enlarged by ViNCEd, since it can handle up to 16 pens instead of the eight ANSI pens:

ANSI Color Activated by

---

0 black CSI 30 m CSI 40 m CSI >0 m

CSI ?30 m CSI 40 m

1 red CSI 31 m CSI 41 m CSI >1 m

CSI ?31 m CSI ?41 m

2 green CSI 32 m CSI 42 m CSI >2 m

CSI ?32 m CSI ?42 m

3 yellow CSI 33 m CSI 43 m CSI >3 m

CSI ?33 m CSI ?43 m

4 blue CSI 34 m CSI 44 m CSI >4 m

CSI ?34 m CSI ?44 m

5 magenta CSI 35 m CSI 45 m CSI >5 m

CSI ?35 m CSI ?45 m

6 cyan CSI 36 m CSI 46 m CSI >6 m

CSI ?36 m CSI ?46 m

7 white CSI 37 m CSI 47 m CSI >7 m

CSI ?37 m CSI ?47 m

The arguments in the range 30..37 set the foreground color, the range 40..47 is responsible for the background color. The sequences of the type "CSI > _ m" select the window fill color. The similar control sequences with the additional question mark "CSI ?_ m" set the color for the user input only.

The following pens are defined for ViNCEd, and are NOT ANSI standard, nor CON: compatible. They work ONLY in ViNCEd windows:

ViNCEd Color Activated by

---

8 grey CSI 38 m CSI 48 m CSI >8 m

CSI ?38 m CSI 48 m

9 light blue CSI 59 m CSI 79 m CSI >9 m

CSI ?59 m CSI ?79 m

10 pink CSI 60 m CSI 80 m CSI >10 m

CSI ?60 m CSI ?80 m

11 violet CSI 61 m CSI 81 m CSI >11 m

CSI ?61 m CSI ?81 m

12 orange CSI 62 m CSI 82 m CSI >12 m

CSI ?62 m CSI ?82 m

13 neon green CSI 63 m CSI 83 m CSI >13 m

CSI ?63 m CSI ?83 m

14 spring green CSI 64 m CSI 84 m CSI >14 m

CSI ?64 m CSI ?84 m

15 dark green CSI 64 m CSI 85 m CSI >15 m

CSI ?64 m CSI ?85 m

This ANSI coloring works ONLY for Kickstart releases 3.0 (39.xx) and up since the necessary system functions are not available in earlier releases.

REMARK: You may alter the ANSI color layout, i.e. the colors that ViNCEd chooses for the ANSI pens. This works either with the private CSI sequence "CSI V" or by using the prefs editor SetVNC , the second "Window" page.

REMEMBER: ViNCEd TRIES its best to find colors that look like the definitions above, but this isn't always possible. Don't expect miracles, on a four color screen screen you'll only see four different colors, not more! If there isn't a red color, then... THAT's IT.

The ANSI colors are mapped to the most similar looking screen pens, regardless if they look "similar" at all. ANSI "red" might get mapped into grey if nothing "more red-ish" is available.

ViNCEd uses shared pens for the ANSI coloring, but tries to allocate more pens if no matching color can be found. Remember that this might be impossible if either the screen has a very low depth or other programs running on the same screen allocated pens as well.

If you want all ANSI colors, or as much as possible, open ViNCEd on a private or public screen, read on here on how to do this. Then define colors to be used by ViNCEd on this screen, by using the preference editor, 2nd window page .

ViNCEd might choose not to allocate additional pens for the ANSI colors if "sufficient" matches are found. This depends not only on if more pens are available, but also on the depth of the screen. ViNCEd is more tolerant on a screen of low depth! For example, ViNCEd usually won't allocate ANY ANSI colors on screens of depth three (eight pens) or below, leaving the additional pens to other purposes. IF you want to use ALL pens as ANSI pens, YOU HAVE TO DEFINE THE COLORS EXPLICITLY, again with SetVNC .

I would suggest to leave the first four colors alone, and fill in the remaining colors to get a perfect ANSI match. If, for example, you're working with the standard CBM workbench layout, a blue, white, grey and black color is already available. Define colors four and up as the missing colors, namely red, yellow, green, magenta and cyan for a full ANSI set, and additionally light blue, pink, neon green, spring green, orange, dark green and violet for a full ViNCEd set of colors.

1.9 VNC Escape sequences

ViNCEd recognizes a big bunch of control sequences. All Commodore ESC-sequences are supported, some VT-220 and some XTerm sequences have been added to complete the set, even some SGI winterm sequences are present. But before, let's clarify some words:

Raw Mode / Cooked mode: Usually, ViNCEd parses the input as lines and on a press of RETURN - this key is also called "Send" in ViNCEd slang - a complete line is send to the input stream, usually the shell. This is called the "cooked-mode". There is also a Raw-Mode (which is, like the name says, "not cooked"). Here, every single character is send to the input stream; this is what usually happens if you open a RAW: window instead of a CON: window. However, there are dos packets to toggle between the two modes. The handling of Ctrl-C to Ctrl-E and Ctrl-Z is rather special in RAW-mode and can be controlled thru a special mode flag, see below.

Scroll Region: While usually scrolling the whole window, it is possible to separate several lines at the top and at the bottom of the window that are fixed, i.e. not scrolled. You might want to put there some global information, like the cursor position, the date, and so on, if writing an editor. The way how these lines interact with the control codes is again controlled by a mode flag. The default way, called "Commodore mode", makes these lines "invisible" to the usual control codes. However, this is not compatible with the VT-220 definition of some control sequences. Therefore, the VT-220 mode can be chosen. This tiny little extra also makes ViNCEd interesting to terminal programs: You do not have to write your own VT-220 emulation.

8 bit mode vs. 7 bit mode: Usually, the Amiga uses eight bit wide characters. However, for some terminal purposes the eighth bit is used as a parity bit and should be ignored for that reason. To reach all ASCII-codes, use in the seven bit mode the ShiftIn and ShiftOut control codes (0E,0F) and substitute the control sequence introducer CSI 9B (155 decimal) by the sequence ESC [. Additionally, this substitution is also made by ViNCEd, so all answer control codes that usually come with CSI are now send with ESC [instead. Your parse code should be smart enough to handle this substitution.

The letter in brackets behind the control codes specifies the origin of a sequence:

(C)=Commodore or standard TTY (2)=VT-220 enhancements added to ViNCEd

(X)=Unix XTerm (S)=SGI winTerm

(V)=ViNCEd (DEC)=Digital

Because the list is so big, it is split into several nodes:

List of control characters

List of unsupported control characters

List of ESC sequences

List of unsupported ESC sequences

1.10 List of control characters.

Here is the list of control codes: (All codes are given in hex)

07 :(C) Bell (blink screen)

08 :(C) Backspace, move cursor backwards. Usually non-erasing, but this can be changed with ESC >?24h

09 :(C) Horizontal TAB. Usually only moving, but can be switched to inserting with ESC 4h

0A :(C) Line feed, scroll window. Scrolling can be disabled with ESC >1h

0B :(C) Vertical TAB = move upwards one line

0C :(C) Form feed: Clear screen and set cursor to origin. If in the Commodore-compatibility mode , and the cursor is not in the border, clear only the scroll region.

0D :(C) Move cursor to start of line.

0E :(C) Shift in: Set bit 7 of printed characters.

0F :(C) Shift out: Do not set bit 7 of characters.

1B :(C) ESC, the control sequence introducer.

7F :(C)(V) A checkered box (a smear character), but can be turned into the forward deleting control character DELete with ESC >?24h

84 :(C) INDex, like LineFeed (0A), but does not place cursor at start of line.

85 :(C) NewLine: Linefeed (0A) plus CR (0D).

88 :(C) SetTab: Create a TAB stop at the current cursor position.

8D :(C) Reverse Index: Move one line upwards (like Vertical TAB 0B).

9B :(C) CSI, the control sequence introducer. Often used below for more complex sequences. Can be replaced by ESC [.

1.11 Unsupported control characters

The following non-printing characters are known by ViNCEd, but will be currently ignored:

8E :(2) SS2: Move to international character table 2 (Standard LATIN-1-ANSI has only the character sets 0 and 1, and ViNCEd does only support eight bit characters)

8F :(2) SS3: Move to international character table 3

9C :(2) Stop DCS sequences (ViNCEd does not support DCS sequences for downloadable character sets)

9D :(2) Start OSC sequence (ViNCEd does not support them)

9E :(2) Start FM sequence (ViNCEd does not support them)

1.12 List of ESC sequences

And now the list of ESC control sequences. Some of the sequences are long forms of eight bit control characters that can be found here .

ESC is the ASCII-character 1B (27 decimal) and SPC is the blank space 20 (32 decimal); n is a decimal number, strg is a text string. BEL is the bell control code 07, and LF the line feed control code 0A (10 decimal):

ESC c :(C) global reset, clear (lower) screen. Reset colors to default, reset TAB stops, select character table 0.

ESC D :(C) like INDex (84), move cursor down one line

ESC E :(C) like NewLine (85), works like LF=0A and CR=0D

ESC M :(C) like Reverse Index (8D), move cursor up one line

ESC H :(C) like SetTab (88), set tab stop at cursor position

ESC SPC F:(2) Switch to seven bit mode

ESC SPC G:(2) Switch to eight bit mode (default)

ESC 7 :(2) Safeback cursor, colors and style

ESC 8 :(2) Restore cursor position

ESC 9 :(V) Restore colors and style

ESC Z :(2) Send terminal ID

ESC] n;strg BEL:(X) Set terminal parameters

ESC] n;strg LF :(V) Set terminal parameters (same as above)

The value of n controls the type of the terminal parameter:

n=0 :(X) Set icon name and window title

n=1 :(X) Set icon name

n=2 :(X) Set window title

n=3 :(X) Set screen title

n=4 :(X) Like n=0 (icon & title)

n=42 :(V) Set keymap. So the command echo "*E]42;d"

selects the german keyboard.

n=50 :(X) Change the window font. The font is given

as "name.size", the appendix "font" must be

dropped. As an example,

echo "E*]50;topaz.8"

selects the topaz font in size eight.

ESC [:(C) Replaces the CSI sequence

The next ESC sequence is only understood by the CSI parser of the SetVNC buffer IO module , don't send this to ViNCEd itself:

ESC @ :(V) Insert the next control character directly.

1.13 Unsupported ESC sequences

Again, there are some ignored or unknown ESC sequences:

ESC P :(2) Introduce DCS control sequence. Since ViNCEd does not support downloadable character sets, this sequence is ignored.

ESC / :(2) Terminate DCS or APC sequence

ESC # 3 :(2) Upper half of double height and double width characters

ESC # 4 :(2) Lower half of double height and double width characters

ESC # 6 :(2) Double width characters

ESC # 8 :(DEC) DEC alignment test (fill window with E's). I considered this one as rather useless.

The following sequences are part of an international character support. This should be done by the locale.library together with the correct font, and not by ViNCEd itself. They are all ignored.

ESC (:(2) Select G0 character font for next character.

ESC) :(2) Select G1 character font for next character.

ESC * :(2) Select G2 character font for next character.

ESC + :(2) Select G3 character font for next character.

ESC | :(2) Select G3 set as GR (LS3R)

ESC } :(2) Select G2 set as GR (LS2R)

ESC ~ :(2) Select G1 set as GR (LS1R)

ESC n :(2) Select G2 characters.

ESC o :(2) Select G3 characters.

ESC N :(2) Single shift to G2 character set.

ESC O :(2) Single shift to G3 character set.

ESC P :(2) Introduce DCS sequence

ESC _ :(2) Introduce APC sequence

ESC ^ :(X) Introduce privacy message

The last three are terminated by ESC /

ESC < :(2) Enter VT-52 mode. (ViNCEd does not support a VT-52 emulation.)

ESC = :(2) Keypad mode enabled.

ESC] 46;arg BEL:(X) Change logfile to "arg". Due to limitations set by the dos library ViNCEd can't support log files.

1.14 List of CSI sequences

The complete set of CSI sequences follow. They are all introduced by the CSI code 9B (=155 decimal), which can be replaced by ESC [. n is again a number which can be missing and defaults to one, unless otherwise stated.

The character code of ESC is 1B (decimal 27), the code of CSI is 9B (decimal 155). SPC is again the blank space.

CSI n :(2) Insert n spaces

CSI n A :(C) Move cursor n lines up. In the VT-220 mode , one can leave the scrolling region with this command, in the Commodore mode this scrolls if the cursor reaches the border of the scroll region.

CSI n B :(C) Move cursor n lines down. Same for VT-220 mode and Commodore mode as above.

CSI n C :(C) Move cursor n characters to the right.

CSI n D :(C) Move cursor n characters to the left.

CSI n E :(C) Move cursor n lines down, set cursor to the leftmost position.

CSI n F :(2) Move cursor n lines up, set cursor to the leftmost position.

CSI n;n H :(C)(V) Move cursor to the specified row and column. n counts from one up (ugly, but true). In the Commodore mode , the cursor cannot be placed outsite of the scrolling region, and the position is relative to the scrolling region.

As an extension, each numeric argument can be preceded by >?, which moves the cursor to an absolute position or relative to the inner scrolling region in the VT-220 mode . Negative values are also accepted to denote positions relative to the bottom/right border.

CSI n;n f :(C)(V) Usually like CSI n;n H, but the details are again a bit tricky. In the Commodore mode , CSI n;n f can be used to place the cursor outside of the scrolling region, which is impossible with the former sequence. In the VT-220 compatibility mode the position is clipped to the scrolling region.

The possibility of adding >? and use of negative numbers works like with CSI n;n H.

CSI n I :(C)(2) Move to the next nth TAB stop. The argument n is an extension and is allowed to be negative to move backwards.

CSI n J : Erase in display

n=0 :(C) (default) Clear starting at cursor

position up to the end of the window.

n=1 :(2) In the CBM compatibility mode, same as n=0.

In the VT-220 mode: Clear from the beginning of the

window up to and inclusive the cursor position,

does NOT touch the lines in the

upper display region . This is a workaround for

a bug in the editor "Ed". (2)

n=2 :(2) Clear all lower lines , do not move the cursor.

n=3 :(V) Same as n=1, but works even in CBM mode like

the VT-220 control code.

In the Commodore mode , the cleared region depends on whether

the cursor is in the scrolling region or in the border. In the

later case, everything is cleared, in the former only the scroll

region is affected.

CSI n K : Erase in line

n=0 :(C) (default) Clear everything under and

behind the cursor position up to the end of the

line.

n=1 :(2) In the Commodore compatibility mode,

works like n=0. Again, this is a workaround for

a bug in the commodore editor "Ed". In the

VT-220 mode clear characters from the start of

the line up to and inclusive the cursor position.

n=2 :(2) Clear the whole line, but do not move

the cursor.

n=3 :(V) Clears characters from the start of the

line to the cursor position, like n=1. But this one

works even in the commodore mode.

CSI n L :(C) Insert n lines

CSI n M :(C) Delete n lines

CSI n P :(C) Delete n characters

CSI n S :(C) Scroll up n lines. In the Commodore mode , this scrolls only the scroll region if the cursor is not in the border.

CSI n T :(C) Scroll down n lines. Again, this will only scroll the scroll region if the cursor is not in the border and ViNCEd is in the Commodore mode .

CSI n W : Cursor TAB control

n=0 :(C) (Default) Set TAB stop at cursor position

n=2 :(C) Clear TAB stop at cursor position

n=5 :(C) Clear all TAB stops

CSI n X :(2} Overwrite next n characters with blank spaces

CSI n Z :(2)(V) Cursor n TABs backwards, n may be negative to move forward.

CSI n v :(V) Safeback mode. Saves back one or more of the mode control bits that can be set with CSI n h and cleared with CSI n l. For the legal values of n, see there. (sort of DEC control sequence that conflicts with another sequence)

CSI n w :(V) Restore mode. Restores one of the mode control bits usually set by CSI n h or CSI n l. See there for more information.

CSI n h :(C)(2)(V) Set mode control flag or flags. Like with CSI n l, CSI n v and CSI n w, more than one flag is allowed here. See CSI n l for all the flags.

CSI n l :(C)(2)(V) Clear mode control flag or flags. If more than one flag is given, the flags must be separated by a semicolon ";".

Here the list of legal (known) values. The default is printed in brackets, but for some mode flags, you may change the default with the SetVNC program as well

>?30 (l):(V) XTerm cursor mode. If this mode is enabled, the cursor won't move when you mark blocks or use the scrollers at the edges of the window. Instead, the window will "pop back" to the original position whenever you type a key or something gets printed. This is the way how XTerm behaves. This flag can be set by the prefs program.

>?29 (l):(V) Special key parsing disable. If set, some keys of ViNCEd are disabled: History,Break (Ctrl-C thru CTRL-F),Stop (CTRL-S), Expand (TAB or Ctrl-TAB), Send (RETURN), Quit. Function keys remain intact. This is useful for writing an editor. Do NOT send in a shell, or you are lost since RETURN is no longer working.

>?28 (l):(V) SendInsertSwap. If set, Send=RETURN and NewLine=Alt-RETURN are exchanged. Useful for writing an editor together with the flag described above. Alone, this somehow emulates the behavior of the MPW-shell of MacOS. More about this can be found in the description of the LINEBREAK argument of SetVNC .

>?27 (l):(V) UserBlockControl. Block operations are not executed, but send as CSI-sequences to the input stream. See below for special codes you can receive.

>?26 (l):(V) RawControl. The break keys Ctrl-C to Ctrl-F and the job control key Ctrl-Z are disabled in RAW-mode. Instead of executing them, their codes are put into the input stream. Does not work in cooked mode.

>?25 (h):(V) AutoPaste. If set, ViNCEd automatically loads a block from the clipboard if the user wants to paste text into the window. If disabled, a request is send to the user. However, marking and copying of text works.

>?24 (l):(V) ErasingBS. If set, the control code BS=08 erases characters and does not only move the cursor. Additional, the control code DEL=7F (127 decimal) works like the keyboard Del-key: It erases forwards.

>?21 (l):(V) BottomAdjust. In turned on, keep the last line of the window adjusted to the bottom of the window if the window gets resized. This is the behaviour of the old CON: window. If disabled, ViNCEd will use its default procedure, that is to insert blank lines or lines from the screen buffer. Also selectable by the prefs .

>?20 (l):(V) Backspace behaviour at start of line. If you press the backspace key at the start of an input line, backspace will erase the line break character, i.e. this line and the line above are merged into a single line. If you set this mode bit to high, this backspace will be ignored. This mode bit is also selectable by default thru the SetVNC program.

>?19 (h):(V) WideScroll. Do not wordwrap at the right border of the window, but scroll. The default value of this flag can be controlled by the SetVNC program.

>?18 (h):(V) FollowPrint. Follow the cursor while printing by scrolling the window to make it visible. The default of this flag can be chosen with SetVNC .

>?16 (l):(V) AutoIndent. After a Send-command (RETURN) or a Split command (ALT+RETURN) place the cursor under the first non-space of the line above. Useful for editors, useless for the shell.

>?12 (l):(V) Shell mode . If set, TAB-expansion (Ctrl-TAB or TAB) and job control (Ctrl-Z) are enabled. The default value of this flag can be controlled thru the preferences editor SetVNC or by the path-option SHELL .

>?10 (l):(V) Edit buffer. If set, the number of display lines is unlimited (well, except by memory).

>?8 (l):(V) ANSI coloring scheme. If set, the "Set Render" sequence CSI m is interpreted as strict ANSI. ViNCEd tries to use the colors that fit best to the ANSI definitions. Read here for more details. This flag defaults to off, but can be set with the preference editor, see here .

Changing this flag yields also a reset of the rendering colors.

>?6 (l):(V) Blinking cursor. The default value of this flag can be set with SetVNC .

>?4 (l):(V) Underscore cursor. Again, the default position of this switch is setable by SetVNC .

>?3 (l):(V) Row mode. If set, the user is not allowed to leave the current line, and ViNCEd is turned into a line editor. Even the RETURN key DOES NOT leave the current line, unlike the RETURN key in the CON: window. This flag can be set by default, see here . Remember: This WON'T BE USEFUL for the shell.

>?2 (l):(V) VT-220 mode . If set, ViNCEd cursor movement is compatible with the VT-220 standard, if cleared the Commodore interpretation is used. This affects mostly the handling of a smaller scroll region and the interpretation of CSI H,CSI f, CSI S,CSI T and some others. Again, this flag can be enabled by default . (V)

?25 (h):(2)(DEC) Display cursor. If disabled, the cursor is invisible. (DEC, to be precise)

?7 (h):(C)(DEC) Wordwrap. If disabled, the text is not broken at the chosen border, but the additional characters are lost.

?5 (l):(2)(DEC) Reverse video. Works only with the ANSI coloring enabled as well. Reverse video will swap the ANSI black and white colors so that the default is black text on white background. The meaning of the ANSI color pens 0 and 7 is interchanged as well. This flag is also selectable by default, see here .

?1 (l):(2)(DEC) Use the numeric keypad for cursor control functions. Can be selected by default with the SetVNC program, see here . More about the functions on the numeric keypad is in this section .

20 (h):(C) AutoCR. If set (which is the default), the linefeed control code 0A (=10 decimal) includes the implicit execution of CR, i.e. moves the cursor to the left, too.

4 (l):(2) InsertMode. If set, printing inserts instead of overwrites. Does not change the handling of user input.

>1 (h):(C) Scroll lock. If cleared, scrolling is forbidden.

CSI n g:(2) Tab control

n=0 :(2) (default) Clear the TAB stop under the cursor.

n=3 :(2) Clear all TAB stops.

n=100 :(S) Reset TAB stops to default.

CSI n m :(C)(V) Rendering control. More than one option can be given, separate them by ";". The restriction to ">" arguments does not go to ViNCEd.

n=0 :(C) Plain text, default color.

n=1 :(C) Enable bold.

n=2 :(C) Faint (secondary color, usually white)

n=3 :(C) Enable italic.

n=4 :(C) Enable underline.

n=5 :(2) Enable blinking. This version of ViNCEd
does not support blinking text, it just sets the
text in bold, like XTerm.

n=7 :(C) Reversed (exchange character/cell colors).

n=8 :(C) Concealed (invisible text for passwords)

n=20 :(V) Plain text, default color.

n=21 :(S) Disable bold.

n=22 :(C) Default foreground color, disable bold.

n=23 :(C) Disable italic.

n=24 :(C) Disable underline.

n=25 :(2) Disable blinking.

n=27 :(C) Disable reverse.

n=28 :(C) Concealed off.

n=30..37:(C) Set foreground color to 0..7, or select
ANSI colors 0..7, depending whether the ANSI coloring
scheme is active. Same goes for all color selection
sequences below.

n=38 :(V) Set foreground color to 8

n=39 :(C) Set foreground color to default or to the
value saved with CSI SPC s

n=51 :(V) Special in the sense that this is ignored
by ViNCEd itself, but used by SetVNC for buffer
outputs written with the PUT argument. This sequence
selects user input text.

n=59..65:(V) Set foreground color to 9..15. The
current ViNCEd version can control up to 16
colors, but not more.

n=40..47:(C) Set background color to 0..7

n=38 :(V) Set background color to 8

n=39 :(C) Set background color to default or to the
value saved with CSI SPC s

n=71 :(V) Again a private SetVNC sequence. Selects
printed text.

n=79..85:(V) Set background color to 9..15

REMARK: If you activated the ANSI rendering scheme with the CSI >?8h sequence, the arguments to CSI m to not specify a pen number (as with CON:), but a predefined ANSI color.

More rendering-options:

>n :(C) Set fill color to n, n=0..15. Unlike the

clumsy console.device, this might appear anywhere

in the control string.

?n :(V) All values of n presented above are

valid, but only change the appearance of the

user input.

As a special hack, if no argument at all is given, the text rendering is reset to the defaults. DO NOT DEPEND ON THIS WORKING IN NEWER RELEASES.

CSI n;n r :(2) Set display size.

This sets the size of the scroll region. All lines above and below this region are not scrolled, thus, kept where they are. The first argument is the first line of the scroll region, 1 being the topmost line. The second argument is the last line of the scroll region. At least two lines must be in the region. If the VT-220 mode is enabled, the cursor gets set to the left top edge of the window.

CSI SPC s :(C) Set default rendering. (SPC is the blank space, ASCII 32=0x20). Sending this sequence sets the rendering type and color defaults to the currently active pens and draw style. This default can be restored later by "CSI 0 m", "CSI 29 m" or "CSI 39 m". However, the defaults get lost on a full reset "ESC c" and are then re-read from the DrawInfo of the screen ViNCEd displays its window on. If you enabled the ANSI coloring , the default is white text on black background, as defined by the ANSI standard. (C)

CSI "62;n p :(2) Selects the bit width.

n = 1 : 7 bit mode (8th bit set to zero)

n = 0 : (default) 8 bit mode

CSI "61p :(2) 7 bit mode

CSI !p :(2) Like ESC c, full reset.

CSI n p :(C) Cursor control

n=0 : Disable cursor

: Everything else, including no argument at all:

Enable cursor.

Some Amiga specific control codes follow. They have in common that in the case of a missing argument the default behaviour is re-established.

CSI n t :(C) Set page length to n lines. The remaining part of the window can be used for rendering.

CSI n u :(C) Set line length to n characters. The remaining part of the window will be kept untouched by ViNCEd and can be used to display graphics.

CSI n x :(C) Set left offset to n pixel.

CSI n y :(C) Set top offset to n pixel.

The next control sequences are ViNCEd specials:

CSI reg,r,g,b V:(V) ViNCEd set color sequence

This sequence will change the screen colors of ViNCEd. They work only if the ViNCEd window was opened on its own screen, see here to find out how to do this. These screen colors can be set by default as well, using the SetVNC program. Colors set in the defaults can be reset, i.e. the CSI V can be un-done with the reset control sequence ESC c .

reg is the index of the color register you want to change. Currently, ViNCEd supports only 16 different color registers.

r,g,b: The red, green and blue component of the color to load into reg. The values are given as 16 bit integers, i.e. 65535 is the maximum.

If reg is between 16 and 31, the color register reg-16 is affected; thus, reg=16 and reg=0 set both the background color. However, the red, green and blue components are now given as four bit integers, i.e. 15 is the maximum. This gives a coarser color specification, but the numbers are more readable.

If reg is between 32 and 47, this sequence selects a color for the ANSI pen reg-32, hence overriding the ViNCEd defaults. The color itself is given as 16 bit values r,g,b. Quite the same, but with four bit colors is possible with reg being between 48 and 63, again selecting the color of the ANSI pen reg-48.

A value between 64 and 95 releases the ANSI pen reg-64. This does NOT mean that some colors are reloaded with system defaults, it is just the pen that is released.

More functions have been added to this CSI sequence. Starting with 3.50, it is also possible to select the cursor color with this sequence:

CSI ?reg,r,g,b V:(V) ViNCEd set cursor color sequence

The following values for reg are possible (please note the question mark in front of the register number):

reg = ?1 : Use a colored cursor, use 16 bit color definitions, i.e. 65535 is lightest.

reg = ?17 : Use a colored cursor, specify colors in four bit values, i.e. 15 is highest color intesity.

reg = ?33 : Use a colored cursor, specify 16 bit color values but load the color with a higher precision, i.e. specify a higher priority for the cursor color. This is useful if you want to load the cursor color register with a definite value even though you're working on a screen with very few pens available.

reg = ?49 : Use a colored cursor, define the color in 4 bit values, use higher precision.

reg = ?65 : Use the default COMPLEMENT cursor, free the cursor color. THIS DOES NOT MEAN that the color registers are loaded with the default screen values you found there before you colored the cursor. This does ONLY mean that ViNCEd gives the control over the color register back to the operating system and uses the old style cursor instead.

CSI n Y :(V) ViNCEd window manager control.

This is a replacement for a similar SGI control sequence that does not obey the ANSI rules and hasn't been implemented in ViNCEd for that reason. This control sequence performs some intuition related actions:

n = 0 : Activate this window.

n = 1 : Send window to front.

n = 2 : Send window to background.

n = 3 : Send the screen this window resides on to the front.

n = 4 : Send the screen to the background.

n = 5 : Show the screen title.

n = 6 : Hide the screen title.

The values n=5 and n=6 are only working if ViNCEd opened its window on its own screen, read here to find out how to do this. They make only sense for BACKDROP like windows and send the screen drag bar behind or in front of BACKDROP windows. Hence, this action can't be seen, usually.

1.15 Sequences that return results.

Some CSI sequences not only set some parameters, they also send back some kind of CSI sequence to inform the user of the result. For example, you may ask ViNCEd to send the current dimension of the window and much more...

The list of the answer codes can be found in this node .

CSI n { :(C) Set Raw Events. n is a list of input events that should be send to the input stream. Unlike with CON:, this also works in cooked mode, but if in raw mode, all selected events will be disabled when switching back to cooked mode. This is again a hack, to make old programs working. As a second hack, the window-close input event is always active when switching to raw mode. This was done to make the program "more" working (sigh). If you do not want to receive the window-close event in raw mode, you must explicitly turn it off. A third special event is the timing event: while this event never worked in CON: windows (try it !), it works in ViNCEd. Since your input stream will be trashed with a lot of timing events otherwise, this event gets disabled once received. You have to re-activate it every time you receive a timing event. For more about the input events, consult the RKRM-Devices and read here to find out which type of input you receive.

CSI n } :(C) Reset Raw Events. The selected Raw-Events are disabled again.

For the next control sequences, "n" is a literal lowercase n, and not a number:

CSI 6 n :(C) Send cursor position, relative to scroll region if in Commodore mode, absolute in VT-220 mode.

CSI >?0 n :(V) Send cursor position, but unlike CSI 6n it is send as a string that can be send back to ViNCEd to restore the cursor position later. It is relative to the scroll region in Commodore mode, absolute in VT-220 mode.

CSI >?1 n :(V) Send absolute cursor position. Leads to CSI n;n f in Commodore mode, CSI n;n H in VT-220 mode.

CSI >?2 n :(V) Send relative cursor position (relative to scroll region), leads to CSI n;n H in Commodore mode, to CSI n;n f to VT-220 mode.

CSI >?3 n :(V) Send ViNCEd version string. Leads to CSI version;revision V

CSI >?4 n :(V) Send mode flags. Sends ESC [and a sequence of mode flags (ESC [h, ESC [l) that describe their current settings.

CSI 0 q :(C) Send window borders, compatible to CON:

CSI q :(C) Like CSI 0 q, but a hack to make csh running. You SHOULD NOT DEPEND ON THIS SEQUENCE, it will be removed sometimes.

CSI >?0 q :(V) Send window borders, together with the scroll region.

CSI >?1 q :(V) Send max. printable domain.

CSI >?2 q :(V) Send window domain.

CSI >?3 q :(V) Send window domain without scroll borders.

CSI c :(2) Send terminal ID (like ESC Z)

And some other parameter requesting CSI sequences. Sorry, but I do not know their meaning:

CSI 5n :(2) Answered by CSI " 0 n

CSI ?15n :(2)(DEC) Answered by CSI ?13 n

CSI ?25n :(2)(DEC) Answered by CSI ?21 n

CSI ?26n :(2)(DEC) Answered by CSI ?27 ; 6 n

CSI >c :(2) Answered by CSI >1 ; 10 ; 0 c

1.16 Unsupported CSI sequences

Unsupported CSI sequences (partially conflicting with Commodore sequences, partially depending on 16 bit characters and other stuff I did not want to implement):

CSI n;n s :(2)(V) Set horizontal scroll region. The selected number of left and right rows do not get scrolled. This should work someday like a horizontal counterpart of CSI n;n r.

CSI n T :(X) Mouse tracking control. Conflicts with Commodore scrolling command.

CSI n x :(2) Request terminal parameters. Conflicts with Commodore's Set Left Offset.

CSI n q :(2) LED control. This should be done by audio programs, not by ViNCEd.

CSI n n :(S) (The second n is a literal lowercase n), with n=100 to 107. Send RGB color codes of the palette entries 0 to 7.

CSI n s :(2)(DEC) DEC save parameters. Conflicts with ViNCEd Set Scroll Region, but functionally replaced by CSI v (see above)

CSI n r :(2)(DEC) DEC restore parameters. Conflicts with ViNCEd / VT-220 Set Scroll Region, but functionally replaced by CSI w (see above)

Some standard mode flags are not supported (mode flags are arguments to CSI h and CSI l):

n=6 :(S) Lock scroll buffer

n=9 :(S) Visual bell. This is controlled by the

sound prefs.

n=12 :(S) Overlay mode. (2nd buffer)

n=?2 :(2) USA characters for G0-G3 set, or VT-52

n=?3 :(2) 132 rows mode. Depends on the size of the

window, not under control of ViNCEd.

n=?4 :(2) Smooth scrolling.

n=?6 :(2) Origin mode. (Cursor absolute to scroll region)

n=?8 :(2) Auto repeat. Under control of the prefs.

n=?9 :(2) Send mouse position on key press.

n=?38 :(2) Tektronix mode. Unsupported by ViNCEd.

n=?40 :(2) Allow 80/132 width switching.

n=?41 :(2) Curses Fix. This is Amiga, not Unix.

Should be part of ixemul, not of ViNCEd.

n=?42 :(2) Enable national character set. Not under

control of ViNCEd.

n=?44 :(2) Ring bell on window border.

n=?45 :(2) Reverse Wraparound.

n=?46 :(2) Start logging. ViNCEd does not support

big brother. (-;

n=?47 :(2) Use second screen buffer.

n=?1000:(X) Mouse tracking on press & release.

n=?1001:(X) Hilite mouse tracking.

All tektronix sequences (not listed here) are also not supported.

1.17 Sequences you might receive

Sequences, you might receive thru your input stream. Please note that in 7 bit mode , CSI is replaced by ESC [.

CSI n;n V :(V) Version report. Consists of version and revision in this order.

CSI n;n R :(C) Cursor position. Answer to CSI 6n.

CSI n;n H :(V) Cursor position. Answer to CSI >?0n and others.

CSI n;n f :(V) Cursor position. Answer to CSI >?0n and others.

CSI n;n;n;n r :(C) Window bounds report. Answer to CSI 0q and others.

CSI n SPC v :(C)(V) (SPC is the blank space 20, 32 decimal) Copy/paste report. Received if AutoPaste mode flag CSI >?25 is disabled or block control is completely under user control by the UserBlockControl mode flag CSI >? 27.

n=0 :(C) Paste

n=1 :(V) Copy

n=2 :(V) Cut

n=3 :(V) Hide

n=4 :(V) Select All

n=5 :(V) Copy quiet (copy, but do not hide)

CSI n;n;n;n;n;n;n;n | :(C) Raw Event report. Requested by CSI {. Works also in cooked mode. For more about the values, consult the RKRM .

The next ones can be herd only in raw mode :

CSI n ~ :(C) Function keys and other special keys. Consult the RKRM-Devices about their meaning.

CSI A :(C) Cursor up

CSI B :(C) Cursor down

CSI C :(C) Cursor right

CSI D :(C) Cursor left

CSI T :(C) Shift+cursor up.

CSI S :(C) Shift+cursor down.

CSI SPC @ :(C) Shift+cursor right. (Again, SPC is the

blank space)

CSI SPC A :(C) Shift+cursor left.

1.18 Book references

For a (not so) complete documentation of all CON: control sequences, read

The Amiga ROM Kernal Reference Manual, Volume Devices. 3rd Edition.

Addison-Wesely Publishing Company, Inc.

ISBN 0-201-56755-X

For a (poor) documentation of AmigaDOS and packet types:

The AmigaDOS Manual, 3rd Edition.

Bantam Books (The Bantam Amiga Library)

ISBN 0-553-35403-4

Additional information about the XTerm and WinTerm control sequences was taken from the manual pages and the documentation of XTerm and WinTerm.

The XTerm docs can be found on every well installed unix system, since this program is a standard tool. The WinTerm docs can be found on SGI systems only.

However, the location of these docs may vary from system to system. You usually obtain them by typing

man xterm

or

man winterm

1.19 ViNCEd window title arguments

Starting with release 3.30, ViNCEd supports special control sequences in the window title that are substituted by information about a shell running in the window. Most of them work only in the SHELL mode , but some of them are valid in either case. They can be used for the icon title of an iconified ViNCEd shell as well. This icon title is usually the same as the window title, but can be changed by the prefs editor .

All of these control sequences start with an percent sign "%", i.e. ViNCEd uses the standard C-style for them:

%% : Insert the percent sign itself.

%N or %n : Insert the CLI number of the shell running in this window. (SHELL mode only).

%S or %s : Expands into the current directory of the active shell. (SHELL mode only).

%R or %r : The numeric result code of the last command. (SHELL mode only).

%E or %e : The numeric secondary result code of the last command. (SHELL mode only).

%F or %f : A readable (text) version of the secondary result code. There's a small difference between "%F" and "%f". If the secondary result code is zero, "%F" expands into the string "no error" (or the localized version of this string) and "%f" into the empty string. (SHELL mode only).

%X : The width of the window, in characters.

%Y : The height of the window, in characters.

The last two sequences refer to the physical window size, not to size of the internal ViNCEd buffers.

1.20 ViNCEd Specialities

ViNCEd is written as a shared library, the vnc.library, which is open to all users. A .fd file and the necessary link libraries are included. It should not be too hard to construct an editor out of this.

It is quite tricky to write a handler that does not reside in L:, but is a shared library, and as such, is expungeable. The vnc.library automagically mounts itself as NEWCON: or VNC: if it gets loaded, and it automatically removes the handler from the dos device list if it gets expunged.

Together with each window comes not only a handler process named by the device name, but, if needed, a process that controls the tab expansion and the job control, together with all other actions that are needed to interact with the DOS. A usual handler itself can't do DOS-IO. A second supervisor process is started by the library; the VNCClip.supervisor handles the clipboard and is a smarter replacement for the usual ConClip, which is no longer needed if work with ViNCEd. As a second job, it is also responsible for re-loading the preferences and installing them into the window.

Since ViNCEd does all the window handling by itself, all windows are by default simple refresh windows. This slows down the scrolling if a ViNCEd window is partly obscured. If you are a bit picky about this, you might want to choose the smart open-path option. The same is done by the window flag "R".

ViNCEd knows a lot more DOS packets than the old CON: handler; you may SEEK and FLUSH inside them, and the (unofficial) ARexx packets QUEUE and PUSH.

ViNCEd supports named consoles , which are a part of the unique job-control mechanism. All so called "owners" are mostly independent of each other, may receive different input events , even in cooked mode, and, in contradiction to CON:, also timer events.

Starting with release 3.20, ViNCEd windows can be opened on their own screens by giving special arguments to the open path , making tools like "ScreenShell" useless. The colors, font and the monitor ID of this screen can be setup by the prefs editor since release 3.30.

The newer ViNCEd releases support now special control characters in the window and screen title, to display the result codes and some more information about shells running in ViNCEd windows. Read on here for more information. They work, however, only in SHELL mode.

A lot of settings and the whole control of the library lies in the hands of one program, SetVNC .

The iconification gadget of ViNCEd is now replaced by an object of a public gadget class called "iconifyimageclass" if a public class of this name is available. This should make it easier for GUI patch programs like SysIHack or MCP to replace this gadget by a custom gadget.

Hey, patch authors! Take your chance and implement this image class!

I regard this as the "cleanest" solution that works without patching any system function at all, so please make use of it (and dump KingKong...)

1.21 Remarks about the ViNCEd buffers

The buffer scheme of ViNCEd is rather unique. First of all, it is split into the history , that keeps commands like the usual CON: window. But this is only half of the story: The graphics buffer keeps the text which is displayed in the window. It is itself split into two parts: The upper line buffer and the lower line buffer. The first one keeps lines that are scrolled off the top edge of the window. It keeps usually only a limited number of lines, setable thru the SetVNC program. The lower line buffer keeps the lines visible in the window, and those that are scrolled off the lower edge of the window.

Again, the data in the display buffer is of one of two kinds: First, printed text, produced by output of the shell and other programs. This data will never be send to the receiving program, usually the shell, when the user presses the Send=RETURN key . It would be useless to send meaningless garbage, like the prompt, back to the shell. However, this text can be copied to the clipboard, but even this can be disabled by SetVNC .

Second, user input: This type of text can be re-send to the shell or whatever program by placing the cursor over them and pressing the send key, usually RETURN . Additional, with some Escape sequences the style and color of both kinds of text can be controlled independent of each other.

Except for that, no difference can be seen between the types.

If you want to change the size of the buffers, read about it in the scroller section .

The new 40.14 release of SetVNC provides arguments to load and save the various buffers of a ViNCEd window. The output is saved in or loaded from a plain ASCII file with embedded CSI sequences that are used to describe the various rendering options of the text seen in the display buffer or in the history. To speedup loading, not the complete set of control sequences is understood - only the most basic rendering sequences like "CSI m" are parsed. If you need to insert more complex sequences, copy them just to the console. However, this is much slower than using the SetVNC buffer commands .

Similar functions for loading and saving the buffers have been introduced in the 3.50 release of ViNCEd itself, i.e. SetVNC is actually no longer needed for buffer I/O. You find the buffer related functions in the leftmost Project Menu .

Anyways, here are the buffer related SetVNC arguments:

SetVNC PUT SCREEN file

Saves the contents of the display buffer in a given file, with embedded CSI sequences for the rendering information.

Similar to "Save As..." in the Project Menu , except that no requester is shown.

SetVNC PUT HISTORY file

Saves the contents of the command line history in a file. No CSI sequences are included here.

Again similar to "Save History..." in the Project Menu .

The following commands re-loads ASCII files into the buffers:

SetVNC GET SCREEN file

Load the display buffer contents from a given file. Only rendering CSI commands are understood here.

Quite the same can be performed without SetVNC using the Open... function of the "Project" menu.

SetVNC GET HISTORY file

Loads the command history from a file into the buffer.

This is again a replacement for the Open History function in the project menu.

For more on command line arguments of SetVNC, read on here .

1.22 Block operations in ViNCEd windows

Like with CON: or RAW:, or in any text editor or word processor, you may mark a so called "block" with the mouse.

To do so, move the mouse pointer over the start point, called the "anchor point", press the left mouse button and move the pointer to the end of the text, while keeping the button pressed. Pointing at the borders of the window will scroll the window contents.

Release the mouse button when you reached the end position of the block. Please note: The upper end of the block is part of it, the lower is not.

Press now Amiga-C to copy the text into the clipboard or Amiga-X to cut it, that is, to remove it from the window additionally. If you selected the "Auto Copy" menu item or the "Implicit copy after text marking" check mark in the SetVNC edit page , this will be done for you if you release the mouse button.

The text in the clipboard can be inserted later into every program that handles the clipboard, like a word processor or ViNCEd itself. If you MUST use CON:, for whatever reason, and the ConClip program is loaded, this will also work with a CON: window.

To paste into a ViNCEd window, activate the window and press Amiga-V. The pasted text is handled as user input, so if the receiving program is a shell , the text will be executed as commands. As a shortcut, and to be conformal to the Unix XTerm, it is also possible to position the cursor and insert the text at the position pointed to with the middle mouse button. This button can be replaced by Ctrl-Left mouse button, see here . For special requirements, ViNCEd can be told to ignore the middle mouse button at all. However, the replacement is still working.

Additionally, you may mark with Amiga-A = "Select All" the complete window contents as a block. You should be a bit careful with that, cause this might be a lot of stuff! With Amiga-H = "Hide block" the marked block can be un-marked.

All the block commands are accessible thru the menu , too, if it is added to the window, but the keys Amiga-C,Amiga-X,Amiga-V,Amiga-A and Amiga-H work even without a menu.

1.23 The ViNCEd scrollers

With the two scrollers on the right and top edge of the window you may re-display the text that got scrolled of the visible domain of the window. These lines are not lost immediately, but a special amount of scroll-off buffer is kept by ViNCEd.

This is very useful for programs that tend to trash the display with huge amounts of error messages, like compilers and assemblers, but is the maximum number of lines is reached, the buffer overflows and the oldest - topmost or bottommost - lines get lost.

The right scroller is used to redisplay these lines, the bottom scroller however to show the characters scrolled off the right edge of the window. The number of characters that may overflow thru the right is currently fixed and approx. 240.

The size of the vertical buffers can be set either with the SetVNC program or by using the settings menu . Just type the desired number of lines as a decimal number into the window, mark this number as a block and cut it with one of the very first menu items of the settings menu .

The "Cut History Size" item cuts the size of the history buffer .

The "Cut Display Size" item cuts the size of the lower buffer .

The "Cut Upper Size" item cuts the size of the upper buffer .

The "Display Size", i.e. the size of the lower buffer should be AT LEAST as big as the window and as the "Upper Display Size".

The scrollers itself can only be added to the window if the window borders provide enough room. This can be archived by adding a size gadget to the window, which is done by default. For more about how to do this, read here .

Depending on the XTerm mode , the cursor may or may not move within the display buffer. If this mode is enabled, the cursor will remain at its position, but the window will be scrolled back to its original position if something is typed.

Command arguments of SetVNC to load and save the contents of the history and the display buffer of a ViNCEd window are presented here

1.24 The history

In addition to the graphical window buffer ViNCEd also supports the usual history that comes with many other window handlers, like CON: or ConMan. It is very useful to avoid re-typing the same commands over and over again.

Every time you send a line to the shell or any other program with the RETURN (the so called Send-) key this line is added to the history, and if the history is full, the oldest line of the history gets lost. The number of lines kept in the history can be set by the Settings Menu item "Cut History Size", like the size of the graphical buffers .

You can later on move thru the history, i.e. redisplay old inputs, with the keys Alt+Cursor up or down, or, alternatively, if the "Alternate Movement" item in the settings menu is checked, with the cursor up/down keys. This flag can also be set by default by the SetVNC program.

It is also possible to search a specific input in the history: Type the part of the line you remember and want to retrieve, and position the cursor at the position where the missing/unknown characters should be inserted. Then press "Shift+Alt+Cursor up" to search in direction of the older lines, or "Shift+Alt+Cursor down" for a search in the opposite direction. If a matching line is found, it will replace your search mask. If you get a unwanted match, erase the result with "Alt+Del" and press again "Shift+Alt+Cursor up" or "... down".

Like with the search command, the Alt key together with the cursor keys is only needed if the "Alternate Movement" item is not checked.

Maybe an example: (_ is the position of the cursor)

co_

finds all lines in the history that start with the characters "c" and "o", like

copy vnc.library to LIBS:

copy ConClip to SYS:Trashcan

but not

delete LIBS:vnc.library

On the other hand, if you enter the search mask

an

an place the cursor over the a (the a in this guide is only underlined if you use MultiView V39 and up, sorry), then only the line

copy ConClip to SYS:Trashcan

is found cause it ends with "an". The same line would have been found with the search mask

coan

cause it starts with "co" and ends with "an", and all the missing characters are in between.

If you do not want to bend your fingers, there is another pair of shortcuts for searching in the buffer, to be found in the Edit Menu . "Amiga-F" searches forward, and "Amiga-B" backwards, like the "cursor-up" and "cursor-down" combinations.

Got it ?

See also: The Keyboard , and the menu or SetVNC section about how to change the keyboard layout to use the simple or shifted cursor keys for moving/searching without "Alt".

A new feature in ViNCEd 3.30:

Starting with this release, identical lines do not go into the history. If you enter the "list" command twice, only the first "list" will go into the history, the second one will be ignored, even though it gets executed of course. This works only if no other command between the identical lines has been entered.

1.25 Name of the window handler

In principle, you could mount the handler under every name you like, but if you use the names "NEWCON:" or "VNC:", the handler gets mounted each time you open the library, and will be removed from the DOS device list once the library gets flushed from memory. It is also possible to keep the handler under two different names in the memory, without wasting memory. For example

mount NEWCON: mount VNC:

will mount ViNCEd as "NEWCON:" and "VNC:", with both handlers using the same code. This will be done by the special library startup code!

Additional, it is possible to completely replace the CON: handler (or any other handler, which is both dangerous and useless) by ViNCEd. This is made possible by the SetVNC program and the line

SetVNC mount override as CON

which will replace CON: with the vnc.library handler. If you do so, you can completely remove the "ConClip" command, as the clipboard control is done by ViNCEd itself.

It is recommended NOT to use an assign command for this job, even though this MIGHT work.

1.26 The TAB expansion

If the ViNCEd window is specified as a shell window , either by default or by a given "SHELL" option in the open path , you may automatically expand path names with a press of TAB or Ctrl-TAB .

Which key is used for that feature depends on the setting of the "Expand with TAB" item in the Settings Menu and of the "TAB filename expansion" check mark of the SetVNC program .

To be compatible with older versions of ViNCEd, SetVNC offers a shell argument to turn on the TAB expansion by hand, read here .

With a press of the expand key - either TAB or Ctrl-TAB - the shell argument under the cursor is understood as an incomplete path name; if TAB is pressed over the first argument, it is seen as partial command. The difference is that for path names only the current directory and the DOS device list is searched, but for commands ViNCEd scans additionally the complete path, as setable by the "Path" command, and the resident list of commands, controlled by the "Resident" command.

Since some users tend to "overuse" the path a bit, searching will get slowed in this case. To speed up the TAB expansion, you may tell ViNCEd to search only in the current directory and the C: directory. This can be done either by selecting the "Partial Path" item in the settings menu or by the prefs editor check mark "Search commands in current and C: only".

A new feature has been introduced to the 3.50 version of ViNCEd: The TAB expansion code provides now a cache that holds the recently used directories. The cache is completely transparent to the user, it is filled on the first attempt to expand an argument and is automatically flushed if the directory on the disk gets changed. Filling the cache does not take longer than a usual TAB expansion request, so there's no need and hence no function to disable this cache manually. The only drawback of the cache is that it operates as if the "TrueMultiAssigns" patch is permanently installed, i.e. all directories of a multi-directory assign are read - regardless if you installed the patch or not. The result of the TAB expansion might, therefore, be inconsistent if you run the system without "TrueMultiAssigns".

By the way: The TAB cache uses the "dynamic memory model" of ViNCEd and fragmentates therefore not the memory.

The way how an incomplete argument or command is used as a search mask is comparable to the way ViNCEd searches in the history :

Only those files math the search mask that are "expansions" of the incomplete path or command given, i.e. some characters at the current position of the cursor have to be inserted to expand the "mask" to the full name. Here some examples, ("_" denotes the cursor position):

1.SYS:> co_

matches the commands "copy", "ConClip", as well as the directory "Commodities" if located at the top level of your disk, and the Device "CON:", unless you told ViNCEd not to match devices explicitly .

1.SYS/Prefs:> copy Set_

matches "SetVNC" and "SetVNC.info" if they are in the prefs drawer of your disk, but not the command "SetKeyBoard" in the command path since the first argument and not the command itself has be to expanded. ViNCEd looks therefore not in the complete command path.

The match of icons can be turned off if you don't like this, again with a flag under control of SetVNC which is located in the forth shell page .

Another example:

1.SYS:> SNC

matches the command "SetVNC" cause it starts with "S" and ends with "NC". All missing characters, in this case "eTV", are between them. Does not match the icon "SetVNC.info", since it has the additional characters ".info" at the end of it.

The next example shows how to match devices, even though you've disabled them:

1.SYS:> Am:

will match all devices or volume names starting with "Am". The colon at the end tells ViNCEd that you want a device. Quite the same is possible if you want only directories - place the cursor before a slash "/" as last character, of if you're looking for icons:

1.SYS:> Se.info

this will match again, as an example, "SetVNC.info". ViNCEd is here clever enough to note that you're looking for an icon here, so this works even if icons are usually excluded.

Usually, not only one file or command will match your search request; if this happens, ViNCEd expands partially only, i.e. only those characters are inserted that are common to all found matches. You are then free to add more characters to your search mask and start the search again. This is a sort of unix "bash" or "tcsh" type TAB expansion I prefer.

If however, you choose not to add more characters to the search mask, press the expand key, TAB or Ctrl-TAB, once again and the first matching file will be inserted completely. Every found expansion can now be seen by pressing the expand key over and over again. You can also move backwards thru this list by pressing Shift together with the expand key.

If you do not want the partial expansion and want to see the first matching file on the first press, toggle the "Expand fully" item in the Settings Menu . This option can be turned on by the SetVNC program , too.

A second item, the "Double TAB Requester", in the same menu can be turned on to enable the file requester: On a double press to TAB, a file requester appears and lets you select the wanted file more comfortably. This feature is also controlled by SetVNC , at the same place as the partial/full expansion switch.

By the way: The double TAB time is the same as the double mouse click time and can be changed by the "Input preferences" editor.

REMARK: This file requester works only if the asl.library or the arp.library is available. The first is part of all new Kickstart distributions, the later is public domain and available thru the AmiNet. The TAB requester will be replaced by the reqtools requester if a RTPatch or similar program is installed.

Starting with release 3.19, two new options have been added for the TAB expansion. The "Request if not Unique" flag is in the settings menu as well, but both are selectable by the prefs editor :

You can also get a requester if a unique expansion is not possible, on the first TAB. Check the "Request if not Unique" menu item in the settings menu to get it.

ViNCEd will try to expand the pattern you gave, and if a partial expansion is possible, i.e. some characters can be added to the pattern, ViNCEd will replace the search pattern by a new pattern. This one is, of course, not unique again, but will be a bit longer, being expanded by the characters common to all matches. Pressing TAB again within the double click time will end up in a requester. If no partial expansion can be found, you get the requester on the first attempt.

Example:

Consider you have the "vnc.library" and the "version.library" both in "LIBS:". If you selected the "Request if not Unique" and type

1.SYS:> list LIBS:v_

followed by TAB, you will end up with

1.SYS:> list LIBS:v.library

and no requester, since some expansion is possible.

If you want the requester on the first attempt, regardless whether a partial expansion is possible or not, select the "Expand Fully" option as well. Then a requester will appear whenever no unique match could be found.

A second option controls what will appear in the requester. If you don't check the "List ViNCEd matches, not directories", a standard ASL (or reqtools) file requester showing the directory ViNCEd searched in will appear - as usual. This might, however, not quite what you want. For example, resident modules are not shown, and the requester does not show the complete contents of a multi-assign.

As a second option, ViNCEd can be setup to display its own list of matches, including all resident modules, devices, files and directories. You get it with the "List ViNCEd Matches" option checked.

However, this requester is not without its quirks: Starting with release 3.30, it is clever enough to let you enter directories like in the standard file requester, but if you use the cursor and return key to step thru the requester, you MAY NOT select devices by this way. This must be seen as a asl or reqtools "feature", and can't be corrected within ViNCEd. Another caveat is that you don't get the ViNCEd requester if you're searching within an device, i.e. your search pattern contains a colon ":". This is again due to some asl/reqtools "feature" (don't call this a bug... :-).

Continue here about how to get this ViNCEd requester.

Another TAB expansion flag has been added to 3.30, which controls the order how the resulting matches are sorted. If the "Show matching directories first" flag in the prefs editor is checked, the directories are shown prior to matching devices, unlike the default.

And just another feature was added, for backwards compatibility with KingCON (yes, KingKong is BACKWARDS!). If you press "Shift+TAB" or "Shift+Control+TAB" within an empty line, a complete list of devices will be generated. This can be seen as a short cut for the longer search mask

1.SYS:> :

1.27 TAB expansion arguments to SetVNC

The TAB expansion mechanism is now an integral part of ViNCEd, but for compatibility reasons it can be still turned on by the SetVNC program by the "TABHook" argument family. (The name was chosen cause in prior versions of ViNCEd the TAB expansion was done by an external code segment called the "TAB hook" because it "hooked" into the ViNCEd code.):

SetVNC [quiet] TABHook [install] [remove] [quit]

With "install", the calling window is turned into a shell window and the expansion key is set to TAB .

"Remove" sets the expansion key to Ctrl-TAB . If the window from which SetVNC was invoked was a shell window , the TAB expansion is, however, not turned off.

"Quit" does exactly the same as "Remove" and is only present for compatibility reasons.

The optional argument "quiet" disables the information messages printed by SetVNC . It is NOT legal to redirect the outputs of SetVNC to NIL:, since the output path is needed to find the calling window.

1.28 The Shell Mode, and Icon drop

ViNCEd can be used as a usual console handler, like CON:, but it offers some additional features that are very handy if ViNCEd is used as a console handler for the shell. That features are the file name expansion with the TAB key and a (rudimentary) UNIX like job control with Ctrl-Z .

These specialities would, however, confuse every non-shell program running together with ViNCEd, thus they are turned off by default.

But, as everything, even this can be changed with the "Use shell mode by default" check mark in the SetVNC program .

To enable TAB expansion and job control use the SHELL path option in the window description for a new shell.

If you explicitly want to disable the shell mode, use the NOSHELL path option.

At least, you are then on the safe side for programs using ViNCEd not for shell output!

The shell mode offers an additional feature, the so called "Icon Drop" support. If you drag a workbench icon over a ViNCEd shell and release the mouse button over this window, this icon will "drop" into this window. Its complete path will be inserted as text then. Starting with release 3.30, you may specify additional qualifiers to insert the directory or the name of the icon only. The default setting is as follows:

Hold down the "ALT" key when releasing the mouse button to insert only the name of the qualifier.

Hold down the "Shift" key to insert the path.

These keys can be changed by the prefs editor , of course.

1.29 ViNCEd job control

The 3.xx releases of ViNCEd provide a feature that is completely new to the Amiga, so called job control . More than once it happened to me that once I called a program from the shell, I found that I actually had to call another one. However, the shell window is blocked by the first window and I have to open a new shell. Under UNIX, this problem is solved thru the key Ctrl-Z that gives control back to the shell.

Something very similar does now work with ViNCEd if used as a shell window : In the situation above, just press Ctrl-Z to get a new shell in the same window! To avoid trashing your input, the running process is suspended from the output, i.e. it is stopped when it tries to print to the window. You'll receive a suspend message in this case.

But then, a second problem arises: WHICH shell receives your inputs and which is allowed to print?

First, a new terminology is introduced: There are so called "background owners" and only one "foreground owner". All owners may send output to the window, but only the later receives input. When pressing Ctrl-Z, the new shell is automatically send to the foreground, but the SetVNC program can be used to put a specific owner to foreground or background. Additional, six tiny scripts are included in the ViNCEd archive to give the complete "UNIX feeling". Amongst them, the scripts "fg" and "bg" and should be copied to your "S:" directory as a part of the installation process. "fg" can move specific owners into the foreground, and "bg" moves the current owner into background.

Named consoles are completely OS compatible to OS 2.0 and up, but not to the old BCPL-Dos of Workbench 1.3 and below. Therefore, job control works only with the newer versions of the OS. If someone really needs this to work with 1.3 and below, send me a note and I will write a patch to make this working in the BCPL version of the OS, too.

More about the SetVNC part of Job control is here .

Read here if you want to setup new owners as a program author.

1.30 ViNCEd Owner System

What are owners?

Owners are groups of processes sharing the meaning of foreground and background. A set of processes, called one owner, can be set to background, i.e. set to receive no user input and can't print to the console, while a second is put to foreground to hear the users commands. At least one owner is always present: This is the NULL-Owner, setup once you open a ViNCEd stream.

Owners not only share the foreground/background property, but also one "thread" of the multi-threaded ESC/CSI sequence parser. ViNCEd cannot be confused by garbled ESC-sequences that come from more than one process. (Yeah, like a real file system this part of ViNCEd is multi-threated!)

How are "owners" identified?{UB}

This is done by another new feature, "named CONSOLE"s. If a program opens a new handle to the current console window, this is usually done by the file name "*", or "CONSOLE:" under 2.0 and up. This will classify the opening program as the default owner. To create a new owner that can be distinguished from the default one, use the named consoles. Open a path like

CONSOLE:name

where "name" is some unique name used to identify your process or process group. If not already created, a new owner will be constructed by ViNCEd, and your process will be added to this owner. As a default, this new owner is not put into foreground, to do this use the vnc.library functions "Foreground()" and "Background()". They receive as arguments the stream resulting in the open call.

Which owner do I get when I open a unnamed console?{UB}

This depends on:

1) whether you open your stream with "*" or with "CONSOLE:"

2) The setting of your processes "pr_ConsoleTask"s field.

If you choose the name "CONSOLE:" you get always the NULL owner that is created every time a new ViNCEd window opens - and that is usually under the control of the shell.

If you choose the "*" name, the owner belonging to your tasks controlling console is used. If opening a new owner, put the contents of the file handles "fh_Type" field into "pr_ConsoleTask", and every stream opened with "*" will belong to the new owner.

The following code segment opens a new file, creates a new owner, puts it into foreground and sets the controlling terminal to the new owner:

BPTR stream;

struct Process *myproc;

struct MsgPort *owner;

stream=Open("CONSOLE:myowner",MODE_OLDFILE);

if (stream) {

myproc=(struct Process *)(FindTask(NULL));

owner=((struct FileHandle *)(BADDR(stream)))->fh_Type;

myproc->pr_ConsoleTask=owner;

Foreground(stream);

}

Just for the case you missed that: Creating a new owner does not bring this owner in foreground, and printing to that owner without further preparation will suspend, i.e. halt you process. You have to bring it to foreground FIRST. As a side effect, all other owners are then in background, and hence even the shell can't trash your outputs.

Another remark: The message pointer called "owner" in the code above is not really the ViNCEd-Owner structure, but if you send your packets to that port, they get identified with the right ViNCEd-Owner.

What actually happens when I press Ctrl-Z?

A new named owner with a unique name is created by opening a named console like above, it is send to foreground, and a new shell process is forked with the terminal process set to the new stream.

A second remark: This job control is still in a somewhat "experimental phase". ViNCEd tries to do this as safe as possible, and for that reason it is not possible to interrupt a program with Ctrl-Z if it is in an "unstable phase" of working. The job control never caused a crash on my system, but it is still a bit picky about the "stability" of the interrupted process. If you have better ideas how to do this stuff, have a look at the vnc.library, especially the entry point "CtrlZSuspend(cn,owner)(a5,a4)". Write better code and patch the library, if you can!

1.31 Installation

Installation is done easiest by the supplied installer script. However, if you MUST install by hand, or if you want to find out what is put where, here is the list:

The vnc.library is put to LIBS: It contains the handler. Unlike all other handlers, it is a library and a handler in one.

The SetVNC program is put to C:. It controls the ViNCEd system in various ways and is also a preferences editor.

Warning: Due to a bug in the virus workshop, the SetVNC program is encoded, to make automatic checking possible. To decode it, use the "Decode" program in the "Extras" drawer. Invoke it like this:

Extras/Decode C:SetVNC

A mountlist entry is copied to DEVS:Dosdrivers. If you have a DEVS:MountList file, add the following lines:

VNC:

Handler = LIBS:vnc.library

Priority = 4

StackSize = 4000

GlobVec = -1

#

On request, the following line is added to your startup-sequence:

SetVNC mount override as CON

It replaces the systems CON: handler by ViNCEd.

The SetVNC.info icon is copied to SYS:Prefs, and the default prefs are copied to "ENVARC:sys/VNC-Configuration" and "ENV:Sys/VNC-Configuration", as well as to "DEVS:VNC-Configuration".

The locale catalogs (german) are installed to the locale: drawer as

LOCALE:catalogs/deutsch/VNC/VNC.sys.catalog and LOCALE:catalogs/deutsch/VNC/VNC.prefs/catalog

The specified guide (default VNC.guide) is copied to the selected place and the ToolType GUIDEPATH of the SetVNC icon in prefs is set to this path.

This path is not part of the preferences, instead it's stored here!

Seven scripts are copied to your S: drawer:

"fg", "bg", "SetKeyboard", "SetFont", "More", "history" and "fork".

If an obsolete DEVS:VNC-Configuration file is found, it is deleted.

1.32 Scripts that come with ViNCEd

As a part of the installation process, seven scripts have been copied to your S: drawer. Three of them are part of the job control , and the others replace standard tools usually found in the C: drawer.

S:fg is used for job control and calls SetVNC to put a specified CLI to foreground. The argument is the number of the CLI process that should be put to foreground. It is the same number you find in the shell prompts and which is displayed as a result of the "status" command.

S:bg is also used for job control and sends the current shell to the background using SetVNC . It does not take arguments.

S:fork starts a new shell in the current shell window and sends it to foreground. If an argument is supplied, this command with all remaining arguments is executed in background. Mostly provided as an example how to use named consoles and job control .

REMARK: There's one quirk about the "fork" command: If you run a program with it, its input and output are set to the new "background" owner, but its controlling terminal, which is used to open a "*" file, will remain the current stream. Thus opening "*" from such a process *MIGHT* end in a deadlock situation! This is due to a gap in the RUN command, and some others in the way how the AmigaDOS executes script files. There's currently no way to fix this with just a script, so take this as a demonstration what is possible.

S:SetKeyboard replaces the C:SetKeyboard command and selects the keyboard for ViNCEd windows. The old C:SetKeyboard still works, but it is more a "hack" than a good program. The script uses the recommended way thru documented ESC sequences. The needed argument is the name of the keyboard, e.g.

SetKeyboard d

selects the german keyboard.

S:SetFont replaces C:SetFont and selects the terminal font. The old C:SetFont still works, but is like C:SetKeyboard more a hack and conceptionally not clean. The supplied script uses the documented ESC sequences to do the same job much better and safer. However, this script does not check for proportional fonts and uses them in the window without warning. As a result, the display might look very ugly and the input of commands will be confusing with proportional fonts. Except that, the arguments are identical to C:SetFont. Maybe somebody wants to write a replacement for C:SetFont that takes care of ViNCEd? In that case, contact me .

S:More is a workaround for the CBM More utility. It is assumed that you keep the original More program in the drawer "SYS:Utilities". If your "more" is located somewhere else, please edit the script!

The script frees the window pointer after more has returned, in order to re-establish the iconification.

S:History dumps the history to the screen or to a file and demonstrates the power of the "PUT" and "GET" arguments of SetVNC .

1.33 My address

Thomas Richter

Rühmkorffstraße 10 A

12209 Berlin

Germany

EMAIL: thor@einstein.math.tu-berlin.de

WWW: http://www.math.tu-berlin.de/~thor/thor/index.html

1.34 ViNCEd internals

Some internal information about ViNCEd can be found here. It is mainly for programming and debugging purposes and not intended to be understandable for "the common user". It is also everything but complete, the whole Autodocs are NOT here but in a separate guide still to appear. However, gurus and packets can be found in this guide.

The concept of Owners and Job Control.

Software failures ViNCEd may create.

Supplied DOS packets.

1.35 ViNCEd software failures

As every shared library, ViNCEd can produce as a last help, software failures (so called "Guru meditations"). They all start with the ID

0x1e000000

and above. This might be a bit unusual to a library if you read how these failure codes should be constructed, but it I considered converting ViNCEd into a device as a replacement for the console.device.

Here the list:

0x1e080001 ViNCEd could not change the window proportions to match the fonts requirements. Re-selection of topaz.8 failed.

0x1e080002 ViNCEd found an alian gadget in the windows gadget list and removed it accidentally.

0x1e08000f ViNCEd found an illegal free mem of a dynamical node.

0x1e060003 Qpkt-failure: ViNCEd send a request to the timer device but received a different one.

0x1e060004 The VNCFiler.supervisor received a command from a window it does not control.

0x1e060006 The VNCFiler.supervisor received an illegal command.

0x1e060016 The VNCFiler.supervisor received an invalid message.

0x1e060013 Qpkt-failure: FindCNWindow received not its own message.

0x1e060023 Qpkt-failure: DoAsciiData received not its own message.

0x1e060001 ViNCEd failed to install its input handlers.

0x1e060002 ViNCEd failed to remove its input handlers.

0x1e000005 ViNCEd could not find a owner that has to be removed in its hash list.

0x1e01000e The message queue lacks memory.

0x1e010001 No memory for IO structures.

0x1e01000f ViNCEd failed to create the VNCClip.supervisor.

0x1e018005 Somebody tried to free memory not under control of the ViNCEd memory manager.

0x1e038002 ViNCEd could not open the graphics.library.

0x1e038003 ViNCEd could not open the layers.library.

0x1e038004 ViNCEd could not open the intuition.library.

0x1e038007 ViNCEd could not open the dos.library.

0x1e138007 The PostRemove() procedure could not open dos.library.

0x1e03801e The handler part of ViNCEd failed to open the vnc.library.

0x1e048035 ViNCEd failed to open the input.device.

0x1e048011 ViNCEd failed to open the console.device.

0x1e070000 The VNCClip.supervisor failed to allocate a signal bit.

0x1e070017 The VNCSaver received an illegal command.

0x1e071000 The startup code of the VNCFiler failed.

0x1e071003 The startup message to the VNCFiler.supervisor was illegal.

0x1e071013 The VNCFiler.supervisor received a message that can't be handled safely in its current state. A deadlock might result!

1.36 Understood DOS packets

For the real freak, the list of DOS packets understood by ViNCEd. Some of them should be considered private, however...

(To avoid unnecessary typing, I stripped the ACTION_ prefix, and I renamed some of the packets to the name of the dos routines they are called from. When going thru the list, you might wonder why ViNCEd supports some file handler like packets, Lock, DupLock, Examine etc. These are PRIVATE, to support a file requester on the matched pattern list. DO NOT USE THEM IN YOUR CODE!)

NIL 0 Dummy packet, does nothing.

Die 5 Terminate handler, if possible.

Current_Volume 7 Get the pointer to the device node entry.

Lock 8 Get a private ViNCEd lock for a match list.

UnLock 15 Unlock a private ViNCEd lock. PRIVATE!

DupLock 19 Duplicate a private ViNCEd lock. PRIVATE!

WaitChar 20 Wait a specified time for input

Examine 23 Start traversing the match list. PRIVATE!

ExNext 24 Read the next entry from the match list. PRIVATE!

Disk_Info 25 Find controlled intuition window. Should be matched with UnDisk_Info.

Flush 27 Refresh the window.

Timer 30 Private (returned timer io)

Read 82 Read data

Write 87 Write data

SendID 500 Send pointer and length to ViNCEd main structure.

Abort 512 Abort all pending read and write requests. This also aborts the suspend state of the waiting process.

UnDisk_Info 513 Free window pointer received with Disk_Info.

ListToLock 514 Convert a match list to a special lock. PRIVATE!

Screen_Mode 994 Switch between raw mode and cooked mode.

ChangeSignal 995 Set port to send break signals to.

OpenRW 1004 Open stream for reading and writing.

OpenR 1005 Open stream for reading.

OpenW 1006 Open stream for writing.

Close 1007 Close stream.

Seek 1008 Seek in queued input lines.

IsFileSystem 1027 Find out if ViNCEd is a filing system (No).

SetOwner 1036 Misused to set ViNCEd owner (good name, but still abused)

SetLine 2001 Send data to the keyboard buffer.

PushLine 2002 Send data to the input buffer, place on top of buffer.

QueueLine 2003 Send data to the input buffer, place at the bottom of the buffer.

ClearLine 2004 Clears the user input at the actual line.

ExpandLine 2011 Private, almost like SetLine, but does not cancel TAB expansion.

Bit 31 of the command is set if the packet is ASYNC. Do not roll your own packets, use the library functions AllocAsyncPacket() or SendAsyncPacket().

(The first is NOT used to construct a packet for the second, SendAsyncPacket does the allocation on its own.)

1.37 Thank you, and Credits page.

Special thanks goes to HiSoft for DevPac 2.0 and to the Software - Distillery for the linker BLink.

Thanks to Ernst Besser for continuously testing this proggy over years, and thanks to Oliver Spaniol for lots of useful remarks.

Thanks to Goran Mitrovic for reporting various bugs in 3.17 and some useful remarks. Not all have found their way to ViNCEd yet, but I'm working on it.

Thanx to Rodja Adolph for his useful remarks about the ViNCEd requesters. Here they are, Rodja. Hope you like them...

Thanks for Albert Bertilsson for the great idea to let ViNCEd create its own screens. This has been added to 3.20, Albert! Thanks to Christopher Naas for reporting a bug in the SetVNC program, which has been removed in the 3.09 release, and again for an enforcer hint of 3.20, which has been removed as well. To bad that I don't have a MMU. What a luck that Christopher is out there to report the hits... (-;

Special thanks goes to Nick "NLS" Sardelianos. Most of the ideas new to version 3.30 are up to him, and life will be definitely harder while he's busy with military service. Thanks a lot, Nick! You considered such a lot of options that I was unable to complete the 3.30 in time, and even now some are missing....

Thanks to Aristotelis Grammatikakis, another guy with a lot of useful ideas.

Thanks to Georgia Pristo, for sending DW and reporting a problem with it - plus more useful remarks.

Thanks to Steve Clack and Miles Willmek for additional ideas new ViNCEd 3.40.

Thanks to all beta testers that helped me to find bugs! It has taken a while to complete the 3.30 release, but has taken even longer without your work!

Thanks to Christopher Perver, Jochen Koob, Eric "Parsec" Spåre and Bernardo Innocenti for additional ideas for the 3.41 release and bug reports.

Thanks again to Bernardo Innocenti for continuously "annoying" me about the scroller gadgets until I finally updated the input handler. Thanks to Stefan Sommerfeld for some other hints, especially for the gadget refresh. You may now disable paths from searching - here's your will, even though I don't like it. Why don't people accept new ideas and want still this old KingKong behaviour? Maybe for the same reason why they don't by PCs... :)

Thanks to Kevin A. Brown for additional ideas I added to the 3.50 release, and to Martin Gierich for his comments, his support and his fine "PatchWord" debugging tool I used for beta-testing. Thanks to Frederic Steinfels for the idea with the colored cursor I was finally able to implement.

I do NOT want to thank Commodore for my @#%&!!! computer. (This is my 3rd computer, my 9th mouse, my 3rd fan, my 2nd SCSI controller, my 3rd disk drive, several memory problems drove me crazy, the expansion port is wrongly designed... Which idiot designed this crap? When I found out that commodore went out of business, I really ENJOYED it.)

Thomas

1.38 The ViNCEd story.

1.06 First working release. Named VNC "VeryNewCon". Not all control sequences are parsed, some workaround for console.device. Finished in 1990.

1.12 and less: Many problems with cursor positioning and smart refresh window.

1.18 Line compressor completely rewritten.

1.24 Workbench 2.04 came out. Problems with scrolling under 2.0, still no block operations.

1.25 First working 2.04, works very safe and stays unchanged for a long time.

1.28 Last 1.xx release.

2.00 Almost everything rewritten. VNC gets now a library instead of a resource.

2.01 Removed massive problems with block operations "Cut" "Copy" "Paste"

2.02 First working 2.0 release. Still somewhat beta (beta than nothing)

2.04 Problems with "Ed" and block operations. Internal Hook

2.05 Converting macros to the menu items did not work.

2.07 Marking of the line end was ugly.

2.08 Removed a lot of 1.xx trash.

2.09 Workaround for console.device bug implemented, FormFeed corrected magic Help key.

2.10 Improved the LibInit procedure with useful alerts (-:

2.11 Removed ScreenToFront bug, new AutoSize SetVNC.

2.12 ConClip OpenCount, smart mount, NOICONS bug removed.

2.13 Menus get the right font, SetVNC keeps track of the title bar, removed again some 1.xx trash.

2.14 Added FindCNWindow(), now used in SetVNC

2.15 Priority of semaphores corrected, menu size adjusted for tiny fonts.

2.16 Added Ctrl+Cursor keys.

2.17 Menufunctions now BOOL instead of void.

2.18 DOS commands now send thru owner instead global port. Lot of updates necessary, removed several bugs (sigh!). Break and owner handling now more dynamic, workaround for "more" (close event expected without request).

2.19 Again, handling of the close gadget in raw mode updated.

2.20 Removed bug in erasing complete lines and bug in prop gadget procedures.

2.21 Scroller get disabled; added another workaround: If the DOS moves the cursor, user inputs are valid starting from this cursor position. This simplifies the input of tables, and works somehow like the screen editor of the Atari 800 XL.

2.22 Scroller disabling now much better, again corrected owner handling.

2.23 Lot of problems with owners, gofer does not work (WaitDosPacket is broken). Rewrote owner system, private hash list for owners. Several days of work. Found bug in rexxsyslib (WaitDosPacket).

2.24 And again owner problems with programs that do not open own streams. Sigh. Corrected read & write. Will this stuff work now?

2.25 I knew it: Corrected owner handling once again. Should not make problems again, but who knows? Again removed some obsolete procedures of release 1.xx beta testing.

2.26 Added FilterInput, HandleKeys updated, and Hook procedure. HandleKeys modified for TABHook. OwnerSemaphore added.

2.27 VNCClip.supervisor added for iff clips. ConClip is now obsolete. Had lot of trouble with the library, found some bugs, esp. with memory problems. vnc.library modified again, removed another (old?) bug in internal Copy/Paste. TABHook stack enlarged, VNC needs at least 4000 bytes, esp. if you run it with a debugger.

2.28 Trouble with ActivateWindow under Kick 1.2/1.3. Does not seem to work better due to a bug in the 1.x libs. Again updated the TABHook to make it working with a NULL lock (Now version 1.01). Forgot to remove option from Makefile, which leaded to illegal hunk linkage.

2.30 Support for locale.library. Prefs are now loaded by the supervisor, added OldOs flag. Removed bug in KillSuper, and unnecessary BSS segments. BuildMenuStructs bug removed, can only seen with "MagicMenus". Again problems with the f*cky owners, removed bug in AClose. When will this sh*t work finally?

2.31 Minimal changes in the lib, new hook for AppWindow. Window size depends now on the text overscan. But completely rewrote SetVNC (2.11). Help now by AmigaGuide, own localization, TABHook program is now obsolete and integrated into SetVNC. New feature: VNC is now an AppWindow (icons welcome).

2.32 Added ALT open option, removed bugs in the parsing of double quotes. (Leftover from 1.xx?) VNCNewWindow now own structure. LibInit and LibRemove remain now in forbidden state, all asynchronous processes are done by the supervisor. Failed to close timer device. This release works pretty well now.

2.33 Removed minor bug in FindCNWindow. Could return TRUE if window was not a VNC window. Caused a crash of SetVNC if called thru a AUX: window. Calculation of cursor position wrong due to rounding errors of divisions.

2.34 Concealed mode handling updated - UserType/UserPenPair can't be controlled from DOS any more - you have to use the structure, like it should. SetVNC's TABHook unstands now double quotes and pathes with spaces. Support for directory assigns added.

2.34/2 No new VNC release, but added the double TAB requester to the TABHook code in SetVNC, which is now in release 2.15.

2.35 Support for NewLook menus of workbench 3.0 added, and removed two tiny bugs. The compatibility workaround to console.device had an illegal pointer, and another in the main code which only occurred if the blitter was too slow. Added support for negative width and height in the open path, window flags are now allowed in every position, to make ARexx working. Sort of dump to place flags there, but want to stay compatible, so this is really not my problem. Rewrote routine to start AmigaGuide, had trouble with MultiView (needed PROGDIR:). I hate this program, it is very likely to crash, esp. with animations. Removed another bug in the DOS module: WAIT/AUTO windows that never opened actually got never closed.

2.36 Scrolling got faster, scrolled only used bitplanes like CON: in 3.1. Luckely not too much trouble. Again removed a problem with window activation under 1.2/1.3 OS.

2.37 The pattern of the inactive cursor is now static and no longer in the TmpRas. This prevents some misprints while scrolling of obscured window parts. Updated the edit buffer hack for table entering, parse position gets updated if one erases or adds characters.

2.38 And again trouble with owners. Action_SetPort refused to accept a port which it could not find a owner for. Starting with this release, the official owner who opened the window gets this port. Not nice, but what to do? The problem is, that you may share streams in AmigaOs.

2.39 Internal beta

2.40 Added rebuild delay.

2.41 Trouble with rebuild delay and "more", added scroll delay. Last 2.xx release.

3.00 Finally started working with 3.xx. Rewrote complete parts of code, esp. owner handling. This was too messy anyhow. Added REAL job control to replace this mess. The TABHook gets now an integral part of the library, called thru a vector. Added button gadgets, and nice tiny arrows at the scrollers. Added VNCFiler.supervisor, displays requester. Removed multi directory assign support, this should be part of the DOS and not of every program. Will write a patch later on. The job control thru Ctrl-Z looks really like a bad hack, but works astonishingly stable.

3.01 Added a lot of CSI sequences I found in XTerm, xwinshell, VT-220 and much more. Still a bug in the TAB expansion, does not find the right argument in the command line. I really thought I tested this... Added a workaround for a bug in csh, sends an illegal CSI sequence and expects the right answer...

3.02 Added support for borders. A lot of testing must be done, and all the scrolling procedures must be rewritten.

3.03 Found a bug in one of the line insertion routines. I fix this one now for the 3rd (4th,5th ?) time and it still does not work. Ugly.

3.10 Thought an iconify gadget could be a neat idea and added it. Lots of work! Found a bug in the shutdown procedure of the VNCFiler. sigh! Completely rewrote SetVNC, old code was a mess after five years of fixing.

3.12 Again some bugs in iconification. Added Job control to SetVNC, really forgot I had to.

3.13 Help menu item did not work, and SetVNC did not save the guide path. I though this worked, but... Added "fork" script and the ability to suspend owners, and removed a bug from Foreground and Background lib functions. Forgot to count the NUL.

3.14 Job control in SetVNC updated, using now "other" for hard cases. Probably should add some more options. Foreground and Background library functions now got flags, too good I hadn't made this public. Added again two hacks for compatibility with "Ed". It sends illegal CSI sequences not compatible with VT-220 and XTerm. Removed crashing "Ed", leaves port with unanswered Read request, I have to flush this manually. Thought I had this trouble once before in one of the 2.xx releases and removed the patch cause it looks so ugly. Updated testing of foreign menus. I wonder why intuition did not crash?

3.15 Thought with 3.xx the owner handling finally works. Proved again that I was wrong! ARRGHH! Fixing the "Ed"-problem added another bug. Really silly, but I think THIS TIME I really made it... (hopefully).

3.16 Tiny bug removed from the window closing procedure. Switching the window to RAW: mode canceled the AUTO and WAIT state completely and made it impossible to close such a window. Removed some bugs in job control: Can't send ARexx to background, crashed when disabling a stopped process and some more. Found a bug in a "correction" of a cursor movement procedure. Sigh. Fixed another bug: Graphic mask was calculated wrong if a single character has to be marked in an empty line.

3.17 TabHook expansions are now sorted by type: Devices prior directories prior files. Thanks, Olli, for the tip! The fix was quite easy and done in five minutes.

3.18 Fixed a tiny bug in the Prefs correction routine. An empty macro does no longer end the parsing of macros, but is accepted as a valid entry. Found a tiny bug in the MatchFirst() function and added a workaround. Fixed a bug in the SetVNC program that might cause a DeadLock on few machines. Replaced the arrow gadgets and scrollers with Boopsi images, if available. ViNCEd looks now O.K. with the new 3D look. Tested with SysIHack, SysI2,MCP, Urouhack and works. However, no standard iconify gadget yet. Removed some compatibility hacks for the console.device, since it does not properly ignore unknown command sequences. Iconify gadget is no longer installed if there's not enough room in the title bar. Fixed two additional bugs in the Prefs correction procedure, and removed the creation of a lock to the window on invocation of a macro. Added an additional check in SetVNC to prevent a crash if a GadgetUp message from a non-SetVNC gadget is received. The prefs flags did not set the gadget properties correctly on installation of the prefs. Sigh!

3.19 Fixed a lot of bugs in the block operations with borders activated. Looks like I forgot these to update in 3.00! Fixed another tiny bug in the line compressor code. I added a "have to think about it" mark to it in 3.00, but I never did. Sigh! Added two options for file requesters and a lot of support stuff in the dos interface module. Had a lot of trouble to avoid deadlocks between the VNCFiler and the dos handler and finally found some use of the pr_WaitPkt pointer. Added a new cursor control mode, XTerm mode, since some users requested it. Removed a bug in the set border routine that might cause a harmless guru - ViNCEd waits now until intuition resized the window. This works only in 2.00 and up, using a delay for 1.xx. Removed an unnecessary flicker in this routine as well. Added some options to the menu, and removed some never used. Made the "Smart Close" even smarter.

3.20 Added support for private screens which can be build on request, and fixed a bug that caused SetVNC to crash if no vnc.library was present. Fixed one bug in the guide about the WINDOW path argument.

3.21 to 3.23 These have been beta-releases. They were never available thru AmiNet. One enforcer hit while scrolling should be gone now. The open path argument parsing was a bit buggy and caused compatibility problems with various programs. The "Next Screen to Front" menu item caused hangs with some additional software. The ConMan "L" option was broken and turned on the seven bit mode accidentally. The TAB expansion routine was a bit buggy with expansion of an empty string, and expanded wrongly if the cursor was placed behind the string. The ordering of the "ICONIFY" open path argument was relevant. The ViNCEd TAB expansion requester did not allow to enter directories. The Boopsi routines were buggy and wasted CPU time. Cursor position selection with the mouse locked the RAW window emulation. As you see, a lot of bugs have been found.

3.30 The new final release. New stuff added to this and the previous beta releases: Window title control codes, new CSI sequences , a better ViNCEd expansion requester , better custom screen support by more open path arguments, icon drop qualifiers, default screen mode and font support. A lot of bugs have been removed that I added to the beta releases. The incompatibilities with DW, MWM and other programs are gone - I hope completely. A flag was added to search only the most useful part of the path in TAB expanding file names, plus some other features. Added the "KEEP" path option, updated the dynamic memory manager - faster and uses less memory now. The guide location is now kept in an environment variable as well.

3.31 Improved the ViNCEd requester once more. There's now no reason to use the standard requester any more. Added a hack to allow VirusWorkshop to check the ViNCEd archive without trouble - manual installation is now no longer possible, due to the encoding of SetVNC. Be warned!

3.32 The BOOPSI scrollers got not updated if the number of visible lines changed. This bug was hidden by another bug in earlier releases. Sigh.

3.33 Internal beta: Found another bug leftover from 2.xx in the line compressor. The settings menu item "Expand with TAB" did not work properly. Added the private CSI sequence "CSI SPC s" for CON: compatibility and ANSI colors. Enhanced the window sizing algorithm to make smaller windows possible. Added asynchronous type-ahead, more flags to control the prop-gadgets, better scrolling, hard cursor scrolling stop, path keyboard control sequences, more flags for TAB expansion, shrunk the minimal size of a window and added another check for the window size, updated jump scrolling, changed the order of the buttons. Added numerical key pad cursor control. Again a lot of work was done!

3.40 Removed an enforcer hit of the 3.33beta2 and one duplicate entry in the library functions. Removed another tiny bug in the boopsi handling.

3.41 The "ESC c" control sequence reset the ANSI mode to the settings stored in the prefs, not to the mode set in the window title - removed. The set render command for color nine did not work. Keypad-mode and Alt-movement conflicted. Added CSI sequences and flags to the prefs to set the ANSI coloring to something different than default. The complete color addressing has changed a bit. As a side effect, screen colors set with CSI V WILL re-appear if the window gets re-opened after an iconification. The old 3.40 dropped these colors (unlike what it should do). Fixed another bug with gfx boards: The color of marked blocks was different at marking and re-printing time, due to the uncertain behaviour of the COMPLEMENT drawing mode for gfx boards. If the screen mode is set to CHUNKY, ViNCEd does no longer try to speedup the block marks with COMPLEMENT drawing, but reprints them. This might be a bit slower, though. Removed a bug in the raster mask calculation routine that might have caused graphics trash for marked and scrolled line feeds. Added a flag to keep the bottom of the window always aligned to the bottommost line in the display buffer. Added a flag to ignore icon ".info" files in the TAB expansion unless you ask for them explicitly. The TAB-Expansion cursor placement policy changed a bit, I think to the better. Added a flag to exchange Shift+ALT Del/BS with ALT Del/BS. ViNCEd installs now a no-op backfill layer hook if possible, hence giving a faster and lesser flickering refresh. Added another "Edit" and "Shell" page to SetVNC to keep the new flags introduced to 3.41. Added the "NumL" key function.

3.42 Removed a bug in the gadget refresh. Forgot to handle this correctly if the prefs get updated. Inserted additional explicit gadget refreshes. The ViNCEd input handler reads now mouse move events if enough CPU power is available, resulting in much smoother scrolling. Fixed a tiny feature of the WaitForChar() function. It did not operate like it should if an EOF signal is pending. Fixed! The guide contained a bug :) The explanation why TAB expansion in kshells did not work was wrong. Sigh. Removed a problem from one of the refresh routines that failed to operate with the no-op backfill hook introduced in 3.41. Added the "harder soft reset" with the Shift key hold down.

3.43 The CloseScreen routine for public screens was broken - ViNCEd was from time to time unable to close its own public screen. Added the "Jump to Next Screen" menu item and a flag to disable the C: path searching at all. The TAB expansion flags parsing was a bit messed up. Everything worked, but the flags had an interpretation different from what the guide said. This is fixed now. The font loader was buggy - called the ASCII string to binary converter with a wrong argument. Fixed.

3.44 This one was a public beta release. I rewrote the clipboard handling completely, added the colored cursor and the TAB expansion. Another new feature are the buffer I/O related functions in the project menu.

3.45 Another public beta. Removed quite a lot of bugs of the 3.44, and just another two bugs left over from 3.43: The line allocation routine had a major problem (urgh!) and the color arbitration logic was somehow messed up.

3.50 Final checks: Added another string to the localization, removed a bug in the sizing of the scrollers (problems with MCP now fixed). Removed a bug in the 1.3 support procedure. Changed the prefs style saving/loading that seemed to be unclear to certain people, added another flag to disable ViNCEd's feature to lock the window on a mouse click. VNC: is now a legal path except for the antique 1.2 and 1.3 versions of the Os due to a bug that used to be in the Mount command.

Read also the credits section of this guide, to find out who is responsible for reporting the bugs (-;

1.39 Compatibility

Some programs don't follow the compatibility guide lines formulated in the RKRM . Some of these problems have been fixed by work-arounds I will remove sometimes.

ViNCEd compatibility guidelines

-Do not send any undocumented CSI sequences. If you want to stay compatible with CON:, send only codes documented in the RKRM .

-If you want to find out whether a window belongs to ViNCEd, use the "FindCNWindow()" function of the library. It also provides a pointer to the intuition window. Call "UnFindCNWindow()" to give this pointer back.

-Do not expect that the stream is linked to an open intuition window. It might be closed by a iconification request any time.

-Do not expect your process is in foreground. Printing from a background process might suspend your program. Read here for more about job control.

-Do not expect that you will receive any raw events if you did not call for, but be prepared that you might even ask for them in cooked mode.

-Do not expect that the handler will break lines somewhere. Use the control sequences to select the mode you need.

-Do not poke in the console unit you receive with DISK_INFO. Changes in this structure are ignored by ViNCEd. In special, do not set the the font in that structure and do not set the keymap, even though this MIGHT work as part of a compatibility hack. ViNCEd supports CSI sequences to do that, read here .

Kown problems with programs:

More

Just a couple of bugs in this program. First, it requests a pointer to the intuition window without giving it back. This is common to all "CON:" programs and will disable iconification and closing of AUTO windows. As a workaround, use the "More" script , which is part of the ViNCEd distribution. It will free the window pointer afterwards for you. Read also about the "FreePointer" command line option of SetVNC here .

Second, "more" expects Close Window events without asking for them. I added a fix for that: Whenever a window is switched to raw mode, the close window event is turned on. Sigh.

Ed

Again, the same pointer problem as above. I said this is usual, but not my trouble. Second, "Ed" sends the undocumented CSI sequence "CSI 1K" which should, according to the VT-220 standard, erase the beginning of a line. Instead, "Ed" expects ViNCEd to erase the end of the line. Added a workaround for this, "CSI 1K" is now interpreted in the wrong way in CBM compatibility mode, but works like it should in VT-220 mode. Third, "Ed" leaves a pending Read packet in the stream of ViNCEd. This packet is now canceled each time a Close Event is send to the owner used by "Ed", together with all other packets pending there. Not very nice, in fact, but prevents crashes and I can't think of another solution. Last, to use "Ed" in a ViNCEd window, you should change the scrolling behaviour by some CSI sequences, with "CSI >?18l" "CSI >?19l" and change maybe the block control with "CSI >?25l". Read more about the control codes here .

Additional note: The "cancelation" of the read packet is done with a bit more cleverness starting with release 3.16.

pdksh resp. ksh

This unix korn shell implementation is another program that does not give back the window pointer it receives. Unless other programs, it asks for the window pointer EACH TIME it displays a prompt (to find out if it is reading from a console, I guess),

making iconification completely impossible, even with a "SetVNC FREEPOINTER". The reason is that after freeing the pointer with this command, it asks again for a new pointer if the prompt gets displayed, ARGHH! The implementation of this function in "ixemul" should be really fixed, since there's currently NO solution for this problem. TAB expansion in "pdksh" windows won't work. The reason is the rather strange reading mechanism of the pksh which somehow emulates the timed reads of unix. Instead of sending a read request to the console, an WaitForChar() style mechanism is used. However, these packets aren't accepted by ViNCEd as stable basis for a TAB expansion, as the packet might get answered at a time not fully under control of ViNCEd - and the CurrentDir() of the waiting process might get lost in between. There's currently no way to fix this. A private TAB expansion like in the unix "bash" would be the better and cleaner solution anyways, so I won't support this bad style.

Z (The Aztec Editor)

Well, if you MUST use this editor (maybe you like VI as well... Argh!), here is how: Send "CSI >?18l" "CSI >?19l" to turn off the ViNCEd scrolling. See also the ViNCEd control sequences .

asm (The lattice assembler, old version 5.0)

This program has an illegal control sequence in its title. When printed, ViNCEd inserts 1988 blank lines. This might take a while, but is not dangerous. The new versions work well.

csh (Respectively "NewShell", by Matthew Dillon, Steve Drew, Carlo Borreo & Cesare Dieni, U. Dominik Mueller and Andreas M. Kirchwitz)

Sends an illegal CSI sequence to ViNCEd ("CSI q" instead of "CSI 0q"). I added a workaround for that, but don't expect this to work any longer. Additional, it does a lot of switching between raw mode and cooked mode, which is quite unnecessary, since the editor features of ViNCEd are superior to that of csh - I advice you to use the "-a" flag. Again, since "csh" reads the window pointer and does not give it back, ViNCEd can't iconify this window anymore (different to KingCON, which still allows the user to iconify the window and, hence, crashes the system). A second problem is, also I haven't checked that out, that the internal commands of "csh" are not in the resident list of the DOS (I suppose). For that reason, they can't get expanded by the TAB key. I advice you to place some dummy commands in the resident list or in the C: directory.

CenterTitles Not a real bug, but the window title might look a bit messy for one or two seconds after opening a ViNCEd window with buttons in it. This messed window title will go away almost immediately and is harmless (and due to a bug in CenterTitles which uses the wrong draw mode to refresh the title.)

ScreenShell The incompatibility with this tool has been removed, since the way how ViNCEd handles custom windows has changed as well. For more on that, read the open path arguments section of the guide.

However, the output of ViNCEd with that program might look a bit different from how CON: outputs looked like. This is due to the fact that ViNCEd does not ignore the settings of the screen pens, different from CON: which does not care about the pens at all.

Compatibility problems of ViNCEd with DW,ZShell and MWM should be gone with the new 3.30 release.

1.40 The SetVNC program

The ViNCEd installation procedure copied the SetVNC program into your C: device. It is a multi-purpose program, used for the control of the complete ViNCEd system. It provides...

The Window Settings Editor

to change the window layout.

The Preferences Editor

to match your personal taste.

The Graphical User Interface of the Editor

the description of all gadgets.

Job Control

send jobs to foreground and background.

Shell Arguments

for fine tuning.

Workbench Tooltypes

for icon lovers.

This program is part of the ViNCEd package and under my copyright, but freeware. Read the licence !

REMARK: The SetVNC program uses overlays to minimize memory usage and disk loading time. Please DO NOT make resident and DO NOT crunch! Some virus checkers have problems testing overlayed programs. THIS IS NOT A BUG in SetVNC, but a bug in the virus checkers!

1.41 Change the window settings

If you not want to change the window settings with the menu , or the desired setting is not under control of the menu, you have to use the SetVNC program. It also offers a nice GUI!

To invoke SetVNC , you may...

Select the "Settings..." item in the project menu of a ViNCEd Shell window.

By default, every ViNCEd Shell comes with a Settings button in the title bar, which also calls SetVNC . However, you may choose to remove this button if you need the space...

If the menu or the button is not available, for whatever reason, you may also invoke SetVNC by hand without any arguments :

SetVNC

However, SetVNC can only install its GUI into a ViNCEd window, you MAY NOT call it from an ordinary CON: window for this job.

If you call SetVNC to modify the window settings or as a preferences editor does not really make a big difference, only two gadgets disappears in the GUI; the second difference is that in the preferences mode the global settings are edited in the window, not the window's settings.

Nevertheless you may choose to save the windows settings as defaults for the new windows...

1.42 The Preferences Editor

To invoke SetVNC as a preferences editor, it is easiest do double click the workbench icon "SetVNC" which should be installed in your Prefs drawer. If you want to call this editor from the shell, use the command line

SetVNC modify prefs

This works ONLY in ViNCEd windows, a ordinary CON: window will not work, cause SetVNC can't install its GUI there!

If you want to find out more about the command line arguments of SetVNC , click here .

REMARK: There is no much difference seen whether you invoke SetVNC as a preference editor or to change the window's settings , the GUI changes only very little. But, in the first case, the global preferences are shown in the editor, while in the later case only the actual window settings are displayed.

If you want to find out more about the GUI, read here GUI .

1.43 The Graphical User Interface

Once you have invoked SetVNC either as preferences editor or to change the settings of your current window , a graphical user interface (GUI) is build in the window. Cause the screen is much to small to hold all gadgets necessary to control ViNCEd, the settings is split into several "pages", grouped by function. Some pages are itself split into a front page and one or more back pages.

The page itself is selected with the "card-index" like gadget on top of the window.

Here is the list:

The About page

The Macros front pages

The Buttons page

The Edit front page

The Edit back page

The third Edit page

The forth Edit page

The fifth Edit page

The Shell front page

The Shell back page

The third Shell page

The fourth Shell page

The fifth Shell page

The Window front page

The Window back page

The third Window page

The Timing page

The System front page

The 2nd System back page

The 3rd System back page

The 4th System back page

All pages except the About page come with with a gadget labeled "Help" in the upper right corner of the window. This gadget shows a certain section of this guide if setup properly. If these gadgets do NOT work, please check that the path to this guide, which can be found in the System page is setup properly. If not, correct it with the string gadget on this page, or locate the guide using a requester: Press the gadget left to the file name gadget in the System page .

Additional to the "Help" gadget, each page has a row of four or three gadgets at the bottom, used to accept or discard the made changes. The second gadget from the right, labeled "To Window", is not present if the preference editor mode. Here their descriptions:

"Save" : Save the changes to a place where they will be loaded each time the system restarts; also, use the settings for all windows to be created.

WARNING: Pressing this button DOES NOT change the settings of the windows already open. Their settings are under control of the programs running in them, not of SetVNC!

"Use" : Use the settings for all new ViNCEd windows, but do not save. The settings will be reset once the system gets rebooted.

WARNING: Typing "Use" WILL NOT change the settings of the windows already open! IT WILL ONLY CHANGE THE DEFAULTS FOR NEW WINDOWS.

Pressing either "Use" or "Save" will also change the settings in the currently active window, as if "To Window" has been selected as well. This is new to SetVNC 40.18, simply because nobody seemed to understand how it worked in older releases.

"To Window" : Install the settings in the window which was used to invoke SetVNC. This works only in the window settings mode.

It DOES NOT touch the settings of all other windows nor the settings of windows to be created in the future.

"Cancel" : Aborts the editor and discards the changes; discards also all shell arguments .

1.44 The About page

The about page....

...presents my logo in the middle. (No, that's not me, I don't have the right ears...)

In the bottom, you find

the "About" gadget, that invokes this guide, if setup correctly. If not, locate it in the System front page .

The next line of gadgets are used to re-load previously used preferences into the editor. It consists of four gadgets if SetVNC is called to modify the window settings or three gadgets if called as a preferences editor .

Their meaning from left to right is:

"Last Saved" : Load preferences saved once before.

"Last Used" : Restore the previously used global settings, undo all changes made so far. These settings are automatically loaded by SetVNC if called as an preferences editor .

"Window" : Re-load the settings from the window which invoked SetVNC and undo all changes. This gadget is only present if ViNCEd was invoked to change the window's settings , and those settings are loaded by SetVNC by default.

"Default" : Load the "factory defaults" of ViNCEd.

The gadgets in the last line are present in each page and described here . They are used to install the changed settings somewhere or to discard them.

1.45 The Macros Pages

The macros pages of the SetVNC GUI let you define the macros , invoked by Amiga-0 to Amiga-9,thru the Macros Menu. Since a SetVNC page is too small to present all macros at once, the gadgets defining the macros are split to two pages; the gadget at the upper right edge of the window lets you toggle between them.

If you want to change the contents of a macros, just type it in one of the gadgets; the text in front of them tells you the number of the macro, and thus the key used to invoke it.

To find out more about macros, click here .

1.46 The Buttons Page

Similar to the Macros Page , on the Buttons Page you can define the Buttons , which appear later in the title bar of the window and are expanded like a macro by a press on the gadget.

Unlike them, buttons come with a title, which appears later in the title bar in the window, and a contents, which is the macro assigned to the name.

The top two string gadgets are used to change them, namely the title and the contents.

Since the number of buttons is not fixed, only one button is shown at a time. Use the "Prev" and "Next" button to bring the previous or the following button to the display, where you might want to change it. The last button presented, if the maximum number of eight is not yet reached, is always empty. If you enter text in the string gadgets above, it is appended to the list of buttons.

If you want to remove a button, click the "Remove" gadget on the left side, the insert beneath let's you insert a new clear button BEFORE the button currently seen, if the limit of eight buttons is not exceeded.

However, it is NOT legal to leave a button with an empty title. If this happens, SetVNC beeps the screen and you have to enter a title, or remove the button completely.

If you want to find out more closely how buttons and the related macros work, continue reading here .

1.47 The Edit Page, part 1

The Edit pages let you select some editor related settings. Since there are so many of them, all settings are split onto three pages; the settings on the second page can be found here , and the third one is here .

And now the description of the check marks on the first page:

Unrestricted cursor movements

The use of the backspace key is somewhat limited: You are not allowed to erase the shell prompt, Shift+Cursor left moves you to the beginning of your input and not "really" to the start of the line and Alt+Del only erases your inputs and not the printed text. With this item checked, all these limitations, and some tiny others, are removed.

As a rule of thumb, moving the cursor now works like moving the cursor by control sequences printed by the DOS: More freedom means more responsibility; more a matter of personal taste (but not mine).

This item works like the "Dos Cursor Move" item in the Settings Menu .

CR inserts below at start of line

Determinates what happens if you press the RETURN key at the start of a line, to be more precise: if the the "Send" key, which sends your inputs to the shell, is pressed at the start of a line.

Usually, RETURN (or "Send"...) inserts a line BELOW the current line, but if pressed at the start of a line, a new line is inserted ABOVE. This is similar to what happens in most editors. However, you may want to change this: If the check mark is activated, RETURN always inserts BELOW.

REMARK: The "Send" key can be set to "Alt+Return" by the escape sequence SwapInsertSend "CSI >?28h". See here . This is not provided as a checkmark in SetVNC, since nobody really should want this. It might be useful however in an editor. (Hi to those Apple folks: Your MPW shell has this strange behaviour of Alt-RETURN and RETURN. Hey! That's UGLY! Can't type as fast as I think...)

TAB moves, no insertion

This one is relatively simple: If not checked, the TAB key inserts enough spaces to your input buffer to reach the next cursor stop; if checked, it just moves the cursor without inserting anything.

The function of Shift+TAB is changed similar: By default, it erases enough characters to move the character to the previous TAB stop. if checked, it only jumps to that position.

REMARK: Don't understand "TAB" too literally. I am talking about the keyboard function, not the key itself (huh?)... O.K., next try: If you want to use TAB for file name expansion, this setting controls the "replacement keys" "Ctrl+TAB" and "Shift+Ctrl+TAB", of course.

Overwrite text

If checked, your inputs overwrite old text; if not, they are inserted into the text already present.

Also present in the Settings Menu : "Overwrite Mode"

And again a REMARK: This does NOT change the insertion mode of a printing program. Use the "InsertionMode" CSI sequence "CSI 4h" to turn on insertion, and "CSI 4l" to turn it off again. Read here .

History buffer wraps around

Controls the way the history keys "Alt+Up" and "Alt+Down" work. If the end of the history is reached, either the "top" old end, or the "bottom" new end, usually an empty line is inserted and further pressing of the same key leads to nothing. If checked, the history "wraps around" and to continue moving thru it at the opposite end.

Also found in the Settings Menu , but differently named: "Wrap Around Buffer"

Don't take "Alt+Up" and "Alt+Down" again to literally. If you choose the "Swap ALT+cursor and cursor keys" in the next page , you move by the un-Alt-ed cursor keys in the history.

By the way: This flag affects also the TAB expansion . If this is checked, the list of possible expansions to a file name will also "wrap around".

1.48 The Edit Page, part 2

This is the second page used to control the editor flags. The first one is here .

Don't write printed text into text into clipboard

Usually, all marked text goes into the clipboard on a Cut (Amiga-X) or Copy (Amiga-C) operation (see Menus), even program output you never typed, like the shell prompt or the directory printed by a "list" command. If this check mark is set, only your inputs are put into the clipboard, the remaining outputs are discarded.

This check mark does the same as the "Cut Inputs Only" in the Settings Menu .

Implicit copy after text marking

Once you marked a block of text, you have explicitly to Copy it with Amiga-C to move it into the clipboard. But even this extra key press can be omitted if you toggle this check mark; if this gadget is checked, the text is immediately copied into the clipboard once you have released the mouse button. This gives a sort of Unix feeling: Copy with the left mouse button, re-insert and place cursor with the middle.

Uh? You don't have a middle mouse button? No matter, holding down the Ctrl-key and pressing the left mouse button does the same! See also the keyboard section of this guide.

Even better, this setting is also in the menu . Here it is simply called "Auto Copy".

Swap ALT+cursor and cursor keys

If not checked, use the cursor keys to move thru the window, and ALT+cursor to move in the history , i.e. redisplay old inputs. However, I found out that some strange guys prefer it the other way round, cause they have still this ol' CON: feeling in their fingers. O.K., you asked for it, so you get it! Toggle this gadget, and the cursor keys up and down are responsible for the history, and ALT plus the keys move in the window.

By the way, there is also a menu shortcut for this: "Alternate Movement"

Underline Cursor

If checked, your cursor is no longer a block with the font's base line marked, but a tiny underline mark. If you prefer it this way, toggle the gadget!

Guess what: This is NOT in the menu, but can be changed by the CSI sequence "CSI >?4h" or "CSI >?4l"

Blinking Cursor

Not that you can turn your cursor in an underline, you can even make it blink - turn this gadget on, and you get it (it, being the blinking). I think it is rather annoying, but it's your taste.

By the way: The speed of the cursor blinking can be adjusted as well. Have a look at the Timing Page .

This is again not under the control of the menu, but a CSI sequence is present. It is documented here , and is "CSI >?6h" for enable and "CSI >?6l" for disable.

1.49 The Edit Page, part 3

This is the third page used to control the editor flags. The first one is here .

XTerm cursor mode

If this one is enabled, the cursor won't move if you scroll the window with the vertical and horizontal props at the edges of a ViNCEd window, or if you mark a block with the mouse. The same goes for the middle mouse button: It inserts the contents of the clipboard to the cursor position, but does not set the cursor, unlike in the usual (native ViNCEd) mode. (More about the middle button and how to substitute it is here .)

The window will be scrolled back to the cursor position if a key is pressed or a program prints to the window. Thus, ViNCEd behaves like the unix terminal program XTerm and no longer like an editor.

This mode can be controlled with a CSI sequence as well, use "CSI >?30h" to turn it on, or "CSI >?30l" to turn it of.

Disable middle mouse button

This gadget can be checked to tell ViNCEd to ignore the middle mouse button completely. This is useful if you want to use this key for a different purpose. However, the replacement key Ctrl+Left mouse button is still working. For a reference, read the keyboard and block operations links.

Disable BS at start of line

This controls the result of pressing the backspace key at the start of a line. By default, ViNCEd removes the line breaking, i.e. removes the current line and appends it to the previous line, like in a word processor. If you toggle this gadget, the backspace key is ignored in this case and no action at all is performed.

This mode can be turned on with the CSI sequence "CSI >? 20h" as well, use "CSI >? 20l" to turn it off again.

ANSI colors as default

Activates the ANSI coloring scheme by default. In this case, ViNCEd does not read the arguments of the set rendering sequence CSI m as pen numbers, but as color indices as defined by the ANSI standard. The default rendering will be white text on black background.

This mode can be turned on with the mode control sequence CSI >?8 h , or can be controlled by the ANSI open-path argument as well.

Don't scroll into the border

Prevents the cursor from moving into non-existing lines. By default, ViNCEd will create empty lines in this case, hence expanding the buffer. But with this flag enabled the cursor will stop at the boundaries of the line buffer .

1.50 The Edit Page, part 4

This is the fourth page used to control the editor flags. The first one is here .

Invisible type ahead

If this flag is enabled, user input will not appear in the window unless a program asks for it. For example, typing another command while the output of the "list" program is printed will not halt and trash the output. Instead, the typed characters will appear later on in the command line. This type-ahead does of course not go for special emergency keys that must be available any time, like Ctrl-S,Ctrl-Q,Ctrl-C to Ctrl-E and Ctrl-Z.

Enable scrollers in RAW mode

The horizontal and vertical scrollers are usually disabled if a window is set to the raw mode , to prevent the user from leaving an input mask setup by the user program. However, this might be inconvenient if ViNCEd is used with a custom shell that switches very often between raw and cooked mode, and will make the lines scrolled off the window unaccessible to the user. With this flag enabled, ViNCEd will keep the scrollers active, hence allowing you to reach other parts of the display buffer . However, a full cursor movement is still impossible since the cursor is completely under the control of the program using ViNCEd. You may scroll forwards and backwards, but are not allowed to leave the current line, hence resulting in an implicit XTerm mode for RAW windows.

Inverse ANSI coloring

This will select the inverse video mode for ANSI coloring , i.e. will exchange the meaning of ANSI black and white. This results in black text on white background as the default ANSI coloring, giving a more readable and eye-friendly appearance.

This flag is also selectable by a DEC control sequence , namely "CSI ?5 h".

Numeric keypad cursor control

Selects the numeric keypad for cursor control functions, like cursor movement and others. More on the precise keyboard layout with this flag turned on is here .

The DEC control sequence "CSI ?1 h" is available to control this flag by a program.

Mouse clicks do not block

If you click into the window while some program is printing and release the mouse button again, the output is usually blocked cause ViNCEd expects that you want to enter text. You may, however, tell ViNCEd that it should resume printing immediately instead of waiting for input. This might, however, mess up the output a bit because it will resume whereever you left the cursor!

This mode is most useful together with the XTerm cursor mode and hence automatically enabled by the XTerm cursor.

Without the XTerm mode, it's propably more convenient to use the keyboard-sequence Ctrl-Q instead to re-enable printing manually, so to leave this flag alone at all.

1.51 The Edit Page, part 5

This is the fifth and last page used to control the editor flags. The first one is here .

Exchange A+BS/Del with S+A+BS/Del

This swaps the meaning of two "Erase" keyboard combinations. With this flag turned on, "Alt+BS" will erase the start of the line up to the cursor position and "Alt+Del" will erase the characters behind the cursor. The old functions "Erase User Input" and "Erase Word" are mapped to "Shift+Alt+Del" and "Shift+Alt+BS", respectively.

Keep bottom of window adjusted

If the window is enlarged, try to keep the bottommost line at the bottom border of the window instead of inserting blank lines. This makes ViNCEd behave like the old CON: on resize. The "CSI >?21 h" sequence exist to control this by an external program, see here for the complete list.

1.52 The Shell Page, part 1

Like for the Edit Pages , there are too many shell related settings to be presented on a single page. This is the first page, and the second is here . Starting with 3.19, a third page has to be setup, which is here . A fourth page has been added for 3.30 which controls qualifiers for the icon drop feature.

They can be reached in the SetVNC program thru the gadget in the upper left corner, which toggles between the pages.

Call macro to close window

This says ViNCEd how to tell the underlying program to quit. If you run ViNCEd not as a shell terminal or have not toggled this gadget, ViNCEd just sends an EOF (End Of File) condition to the program. The shell of the Workbench 2.0 and up is clever enough to do the proper thing: Shut down. However, this goes not for the old BCPL shell of Workbench 1.3 and below. Very politely, it does nothing but trashing the memory. To prevent this, a smarter thing could be done when the user wants the shell to close the window - ViNCEd sends its Quit shell , which is defined in the System Page , and usually contains an "EndCLI" command. But even with a newer shell, sending a macro instead of an EOF might be the cleverer choice, you could, for example, call a cleanup script before quitting.

To make it short: A macro is called to close the window if this gadget is checked, else the usual EOF is send.

Starting with 3.19, there's a second macro that is called to close the window. It is called "Quit program" and used to shutdown the window if a command is executed in the shell. It is defined in the forth System Page of the SetVNC program .

REMARK: Since sending an EOF is dangerous in Workbench 1.3, this gadget is NOT available with the old version of the Workbench. Same goes for the menu item in the Settings Menu , which is called "Smart Close".

And a second REMARK: Closing the windows works also with the key "Ctrl-\" (backslash), but when pressing this key ViNCEd ignores the "Prevent accidental window closing" checkmark and REALLY tries to close. Read the next description to find out what this thingly means.

Prevent accidental window closing

It happened more than once to me that I pressed the close gadget by accident, but since another process was running on the same window - usually an editor - I was left with an ViNCEd window without a shell. To prevent this, toggle this gadget and ViNCEd will ignore the close gadget if more than one process runs with this window as input or output. If you want to close the window explicitly in this situation, you have to use the "Ctrl-\" key .

This setting is also available in the Settings Menu , it is called here "Safer Close".

Don’t close until everybody waits

This one is both, tricky and technical, and you shouldn’t care about it usually. Here is what is means:

Consider you opened ViNCEd with the AUTO - open option . This means that the user may temporary close the window, if no process prints on the window and wants to receive input. If a process DOES listen, i.e. waits for input, the close gadget will finally close the window.

This gadget now defines what happens if more than one process uses the window: If the gadget is not checked, and one of the process is ready to receive input, the close gadget sends the Close Window Macro or an EOF, i.e. tries to shut down the window. If the check mark is turned on, ViNCEd only closes the window temporary and reopens it, if one of the processes is doing output or requests additional input. The final window close procedure is only engaged if ALL processes are waiting for input.

This strange setting is also found in the settings menu, under name "Auto Close Queue"

Don’t add close gadget by default

Decide whether every ViNCEd window should come with a close gadget or not. The tradition of CON: windows is not to install a close gadget, but the ViNCEd tradition is different: It usually installs a close gadget into its windows. Check this gadget, and you do not get a close gadget by default, release it for the ViNCEd default.

The setting of this gadget is overridden by the "CLOSE" open option , which is given in the open path for a ViNCEd window. With "CLOSE" you can explicitly ask for a close gadget, with "NOCLOSE" you can suppress it.

Don’t add iconify gadget by default

Choose if you want an iconify gadget on every ViNCEd window that opens on the workbench screen. If checked, you have explicitly to ask for it with the open option "ICONIFY", if not, the iconify gadget comes by default to all windows opened on the workbench and you have to suppress it explicitly for these windows with "NOICONIFY" in the path.

If you open a ViNCEd window on a different screen than the workbench, the default is NOT to install the iconifier, regardless of the setting of this gadget. The point is that the icon can only appear on the workbench, not on other screens. If you don’t mind, you have to use the "ICONIFY" open argument .

1.53 The Shell Page, part 2

This is the description of the second page for shell related settings. The first is found here . By the way: You can toggle between the pages by the gadget in the upper left corner.

TAB filename expansion

Does actually not turn on and off the filename expansion , but selects the key that is used for expansion in shell windows . TAB expansion won’t work for safety in non-shell windows; you can’t select with this gadget, whether a window should be considered as a shell window or not, this is done by a switch below.

If checked, TAB and Shift+TAB are used for expansion , if not TAB is the usual tabulator key and expansion is done by Ctrl+TAB and Ctrl+Shift+TAB.

This setting can be changed from the menu by the "Expand with TAB" item in the Settings Menu .

Double TAB requester

Select whether you want to have a file requester on a double press of the expand key , selected with the gadget above, or not.

If you checked this gadget and you press later on the expansion key TAB or Ctrl+TAB twice in the double click period, a file requester will pop up and let you select the desired expansion with the mouse.

By the way: The double click period is set by the Input prefs program in your systems "Prefs" drawer. It is used for double clicks with the mouse as for the double TAB’s.

This item is also available by menu. There, it is "Double TAB Requester"

First TAB expands fully

Decide if the first press to the TAB key (or Ctrl+TAB) expands to the first file name matching the pattern, or if the pattern should be expanded only partially into the best matching pattern that fits all found files; you have then a chance to give the pattern more precisely, or press TAB a second time to do the full expansion .

But, as I said above, the whole expansion stuff works only in shell windows .

This function can be controlled by the menu item "Expand Fully", too.

Requester if expansion is ambiguous

Tells ViNCEd to setup a requester if no unique match can be found in for a TAB expansion pattern. Thus you need only one TAB to see the requester, unlike two with this option disabled. By default, this option DOES NOT invoke the requester if a partial expansion is possible, i.e. characters common to all matches can be added to the pattern to make it more specific.

If you want a requester in either case on the first TAB and don't want a partial expansion, turn on the "First TAB expands fully" option right on top of this one.

This option is also in the settings menu , it's called "Request if not Unique" there, to keep the name short.

For more examples on this, read the TAB Expansion chapter of the manual.

Show ViNCEd matches, not directories

Ask ViNCEd not to setup a standard ASL (or reqtools) requester on the directory it looked for expansions, but to display the list of possible expansions itself. This list includes resident modules, multi-directory assigns or devices, if you looked for them. However, you won't be able to go into found directories or devices from this requester, you're just able to insert them.

For more quirks about this option, read on here .

NOTE: The "Use shell mode by default" option has moved to the next page .

1.54 The Shell Page, part 3

This is the description of the third page for shell related settings. The first is here .

Use shell mode by default

Select if all windows shall be shell windows by default. This has several consequences, for example enables the TAB expansion and the job control , as well as the menu . If this gadget is checked, you don't have to give the "SHELL" open option to get a window for a shell, but it is more dangerous since you have to use the option "NOSHELL" for all programs that only want a usual window. To use TAB expansion or, even worse, job control in a non shell window could be rather fatal.

For that reason, I ADVICE YOU NOT TO USE THIS OPTION if you replaced CON: with ViNCEd, since many programs will use CON: without wanting a shell. See also the arguments of SetVNC.

This flag DOES NOT switch the shell mode of the current window, it just controls the default if neither a "SHELL" nor "NOSHELL" option is given in the open path ; especially changing this flag only in the current window settings leads to nothing, cause the mode of the window is already selected - to turn a ViNCEd window into a shell window, use the "SHELLMODE" argument of SetVNC.

Search commands in current and C: only

This affects the way where ViNCEd searchs for commands in the TAB expansion .

If you expand the first word on the current line, ViNCEd assumes that this is supposed to be a command, thus searches the complete path. However, some people tend to overuse this path by adding tons of directories to it and thus slowing down ViNCEd considerably. By setting this flag you tell ViNCEd to ignore this path and look for command expansions only in the current and C: directory, hence speeding up the TAB expansion.

There's now a second flag determinating the TAB expansion path which interacts with this one. It can be used to cut down this path even more. Read on on the next page .

Show matching directories first

The ordering of the found TAB expansion matches is changed: Directories are shown first, then devices, then files. The default is to show devices first.

Please note that this affects the ordering in the TAB expansion list only - hence what you see first if you step thru this list manually with the expand (TAB) key or the reverse expand (Shift+TAB) key. This DOES NOT affect the ordering of the matches in the requesters. It is up to the asl/arp/reqtools library and can't be changed from within ViNCEd. However, the reqtools library has a switch how to order the files in this requester, so consult the reqtools documentation in this case.

Devices don't match implicitly

If this is turned on, devices won't be added to the TAB expansion list unless you ask for them explicitly, by giving a colon in the last part of the TAB expansion pattern.

Enable close requester

If you set the "Safer Close" menu item or the "Prevent accidental window closing" flag in the prefs editor , this item can be checked to get an requester in the case that you want to close a window with more than one process running in it. You may then choose to close the window anyways, or to ignore this close signal.

Again, this flag is USELESS if you haven't selected the "Safer Close" or equivalent flag I mentioned above!

1.55 The Shell Page, part 4

This is the fourth page for shell related settings, again dedicated to the TAB expansion . There's currently only one flag here:

Icon files don't match implicitly

Prevent ViNCEd from inserting ".info" files, i.e. workbench icon files into the TAB completion list unless you ask for icons. IF you want icons, search with a pattern like

1.SYS:> list .info

where the underscore marks the cursor position.

Shorter command search paths

Cut the path for TAB expansion even shorter if the first argument is to be expanded. With this gadget checked the TAB expansion search path for commands defaults to "C:", the current directory and the resident command list. If you enable the "Partial Path" as well, the command search path is cut down to the current dir only, as for all other arguments.

1.56 The Shell Page, part 5

This is currently the last page for shell related settings. The fifth shell page is used to setup special qualifiers for the icon drop feature.

If you press one of the keys listed on this page while you're dropping an icon of the workbench in the ViNCEd window, only the path or the name of the icon will be inserted.

The left hand gadgets control which keys must be pressed to insert only the name of the icon, the right hand side is used to specify the keys for inserting the path only.

Please note that ViNCEd does not distinguish between the left and right Shift or Alt key, each of them will work.

More than one gadget can be checked if you want to enable these functions with pressing more than one key at once. Please avoid selecting the same qualifier on both sides, or selecting no qualifier at all. This is valid for ViNCEd, but may result in disabling the regular icon drop feature.

1.57 The Window Page

This page is used to setup the size of the buffers used by ViNCEd, and some window related settings.

This page used to be called the "Buffers Page" in previous releases of ViNCEd, but for ViNCEd 3.30 a second page has been added which is not buffer related.

History lines

Enter here the size of lines kept in the command history . It holds commands you entered in the window and lets you redisplay them with the Alt+Cursor or Cursor Up/Down keys . If the history overflows, the oldest input gets discarded.

Which keys are used depends on the setting of the "Swap Alt-cursor and cursor keys" setting in the edit page .

Values from 8 to 1024 are valid.

Upper display size

Choose the number of lines hold in the upper display buffer . This buffers keeps lines that are scrolled off the top edge window. You can redisplay them by using the scroller on the right side of the window, or by moving the cursor upwards with the Cursor Up key, like in a usual editor. If you checked the "Swap Alt-cursor and cursor keys" setting in the edit page , the cursor will be moved with Alt+Cursor Up instead.

As a rule of thumb, the upper display buffer should be half the size of the Lower display buffer, or bigger. The minimal possible size is 8, the maximal is 1024. If the upper display buffer overflows, which is usual when scrolling big directory lists, the topmost line gets lost. So if your listing does not fit, enlarge this buffer.

Lower display size

As you already might have thought, this is the number in the lower display buffer , which keeps the lines scrolled off the lower edge of the window and the text in the window itself. This buffer should be AT LEAST as big as the number of lines visible in the biggest possible window, values can range here from 128 to 1024. But I advice you to choose the size of the lower buffer at least twice the size of the upper buffer, since many lines go in this buffer if you scroll backwards to see the top of the window. If the buffer is too small, your latest outputs will be lost when scrolling back to the bottom of the text.

Monitor

This gadget is used to setup the default monitor ID for custom screens . The left hand button gives a screen mode requester if available - please select here your prefered monitor for ViNCEd screens.

The string gadget at the right side must be used if no screen requester is available (like for people still using the 1.3 or 2.0 workbench). Hex notation using "0x" or "$" is valid here, but the upper sixteen bits (the monitor ID itself) are ignored if you run ViNCEd on a 1.3 workbench.

Font

Select here the default font to be used in ViNCEd windows. Pressing the button yields a standard font requester, please choose a good font for ViNCEd here.

The string gadget next to the button is useful for setting up the font manually if no font requester is available. Enter here the name of the font without ".font", followed by a "." and the size as a decimal number. Hence, the string "topaz.8" will select the topaz font, size eight as the default.

This setting can be overridden by the "FONT" argument in the path specified for a ViNCEd stream.

If you leave this gadget empty, the system default font will be used here.

Please note that the screen default font (and hence the font used for the menus) can not be selected with this gadget! This is up to the screen on which ViNCEd opened its window. The screen font of custom screens is currently not under control of ViNCEd and can be changed thru the system font preferences only.

If you want to find out how to setup the colors of the ViNCEd custom screens, read the next page , too.

1.58 The Window page, part 2

This is the second page of SetVNC to control window related settings. Click here to read the first one.

This page is completely decidated for setting up the colors of ViNCEd custom screens or for the ANSI pens. These screens are used if special arguments are present in the ViNCEd path . The colors specified here are then loaded to the this custom screen.

This is most useful if you want an ANSI like color layout. You don't have to choose your colors exactly in the same order as the ANSI defintions, as ViNCEd picks the right colors in ANSI mode anyway. Read on here for more on the ANSI coloring scheme.

To edit one color, you have to select first the usage of this color with the two checkmark gadgets near the bottom of the window:

Load register

Try to load this color into the shown hardware register. This makes only sense on a private screen. ViNCEd ignores the color register if the window is presented on a custom screen.

Even unreserved colors will be used by ViNCEd in the "non-ANSI" mode if you send the proper rendering CSI sequence "CSI m" command, but the actual color that appears on the screen is then up to the system.

You may even change the color on the screen with the "CSI V" sequence later on, but "reserved" colors have the advantage that they can be restored with the reset control sequence "ESC c". Read here for a complete guide about ViNCEd control sequences.

Please note that the hardware registers zero to four colors are usually used to render the window frame, the buttons and the menu as well. Using strange color combinations here results definitely in a "strange look". The "3D"-look will go away, for example. Try to use a dark color for index 1, and a light color for index 2 - or avoid window frames at all using a backdrop borderless window! See the path arguments part of the guide to find out how to do this!

Define ANSI mapping

Don't define a hardware register, but the default color for the ANSI mode. The color index is in this case not the hardware register number, but the ANSI pen number. This flag works also for public and workbench screens, but doesn't load a color register unless no other matching pen was found.

It makes sense to set both flags. ViNCEd tries in this case first to arbitrate the selected hardware register to load the desired color for the ANSI pen of the same index. If this register is not available, ViNCEd tries to allocate a shared pen for the color, but with a higher precision than with only the ANSI flag selected.

The CSI V sequence does, for example, reserve ANSI colors by setting both flags.

The sliders

Checking either or both check marks makes the sliders at the left hand side of the page available. They are used to setup the red, green and blue component of the color. A color box at the right side of the page displays a sample of this color.

"Prev" and "Next"

The "Prev" and "Next" buttons at the right bottom of the page are used to select the index of the color you want to change. They refer to the arguments you give to the rendering CSI command "CSI m", see there to find out more how to select a specific color register. Only 16 colors can be controlled by ViNCEd, so opening a "non-ANSI" ViNCEd window on a screen with more than 16 colors will waste only memory, not more! Please note that ViNCEd is, after all, a terminal and not a drawing tool.

This page is now also used to setup the cursor color if you want a colored cursor at. The cursor color is actually "below" the screen colors, so pressing the "Prev" button with color 0 being shown in the window brings you to the definition of the cursor color. The numerical color index is now replaced by "Cc", to indicate that you're now defining the cursor. Activate the "Load Register" checkmark to get a colored cursor, and specify the color you want for it. The "Define ANSI mapping" gadget can be activated as well, ViNCEd will try to allocate the cursor color with a higher priority in this case, resulting in a better color match especially on a screen with few color registers available. Activating this gadget alone does not make any sense at all, at least not in the current release.

REMARK: It might happen that you can't see a sample of the color beneath the sliders cause all screen colors are allocated by other programs. In this case SetVNC tries to allocate a hardware sprite to display an additional color. This will yield to a rather strange behaviour of this color box. For example, it might "jump" out of the window when moving the window around, or might disappear if you activate a different window. This is due to hardware limitations! It might even fail to appear at all, either if no hardware sprite or no DMA time is available! This might happen with hacked monitor drivers or overscanned screens. Please note that it is hard enough to display an additional color on a four or eight color workbench! No problems should arise on Gfx boards cause they offer usually enough free pens that are used in this case instead.

1.59 The Window page, part 3

This is the third page of SetVNC, used for miscellaneous windows related settings. Click here to go to the first window page.

Disable horizontal scroller

Prevents ViNCEd from attaching a horizontal scroller gadget to each window. The gadget can be added explicitly to windows with the open path argument "PROPX", read here .

Disable vertical scroller

ViNCEd won't add a vertical scroll bar by default if this gadget is checked. You may then ask ViNCEd explicitly with the "PROPY" open path argument to get this gadget.

1.60 The Timing Page

The timing of ViNCEd is setup by this page. This is not only the blink speed of the cursor - if you turned that on - but also two delay rates for refresh and scrolling.

All timings in this page are measured in milliseconds, that is a thousands of a second. The screen of a usual TV is refreshed every 20 or 17 milliseconds, depending on the system, PAL or NTSC. Values can be entered as the number (in "ms"), or chosen with the slider right to the gadget.

Cursor Blink Speed

Choose the blinking speed of the cursor here, if you like this feature. Values range from 100 (very annoying flickering) to 1000 (boredom).

Use this page to turn the blinking on or off.

The next two settings are only important if the "Allow delayed window refresh" gadget at the bottom of the page is checked. It tells ViNCEd to delay the window refresh if the graphics output is too slow. This might be a bit irritating in the first place, but speeds up output a lot. The precise timing depends highly on the graphics subsystem and on the processor, and you should adjust this.

REMEMBER: "Slow" for a MC68060 is something different than for a MC68000!

Vertical scroll threshold

This setting specifies what "too slow graphics output" means in detail. If text can't get printed faster than that, the output is delayed and will be printed when there is enough time. Values are from 25 (things are really in a hurry, MOVE ON!) to 500 (let's take things easy). If you have a slow processor (like the MC68000) or slow graphics (many colors), be a bit more tolerant and increase this value!

Rebuild delay

After text output has been delayed, either because it is slower than the speed specified by the above value, or a horizontal scroll is necessary, this is the time ViNCEd buffers all incoming text and suppresses output. The complete text is printed when the time entered in this field is over. The timing of the scrollers on the edges of the window is controlled by this value, too, cause scroller refresh is slow.

Use values from 100 (no remarkable delay) to 1000 (quite a lot of time passes until refresh), but remember:

If you choose a large value here, a lot of text might get printed, and it might get scrolled out of the window before you see it!

Also, I advise you to choose AT LEAST twice the value of the "Vertical scroll threshold", or timing might get a mess cause delaying the output then won't speed up the output considerably.

Intuition Delay

This setting is a mess, and it's only needed if you work with the old 1.3 workbench (whyever ?). Due to a bug in the old intuition.library, after resizing or moving a window the calling program must wait some time before the window parameters get updated and the refresh statement is set. Even worse, the whole refresh timing is a mess. ViNCEd tries to avoid such problems by waiting some time until intuition does the necessary operations. This time is somewhat processor dependent, and you might try to choose a smaller value if you have a faster computer.

There is no need to touch this if you use the workbench 2.04 or up, since this timing is no more needed then.

Allow delayed window refresh

Toggle this gadget and ViNCEd will use it's relaxed window refreshment, like delayed printing and jump scrolling for slow output devices. The timing of this relaxed handling is then set up by the "Vertical scroll threshold" and the "Rebuild delay" above.

If this gadget is not checked, ViNCEd scrolls like the usual CON: handler and the two values are irrelevant.

This check item can be found in the settings menu , too. It is called "Rebuild Delay" there.

1.61 The System Page, part 1

This is the first page of the four System Pages, that control a lot of internal stuff of ViNCEd. The remaining three pages can be reached by the gadget in the upper right edge of the window.

ViNCEd Guide Path

Enter here the COMPLETE path to the VNC.guide that came with the ViNCEd archive (and you are currently reading). The gadget left to the path name will bring up a file requester, to help you locating the guide. However, this requester will work ONLY, if the asl.library or the arp.library is available. While the first one comes with the Workbench, the last is public domain and is used as a replacement for the old workbench releases.

If you want to make the online guide system working, please make sure that:

1) The path of the guide is correctly setup in this gadget

2) The default tool of the guide itself is setup properly. To change it, locate the guides icon on the workbench, click it ONCE with the mouse and select "Info" from the workbench's icon menu. Enter as "Default Tool" in the upcoming window the COMPLETE PATH of your favorite guide browser; this should be "SYS:Utilities/Multiview" for Workbench 3.0 and up, "SYS:Utilities/AmigaGuide" for all other systems. "Multiview" comes with version 3.0 and 3.1 of the workbench, the "AmigaGuide" can be found in the AmiNet, for both versions of the OS, 2.0 and 1.3. However, the last one is a bit buggy (I had some crashes while testing it), and I recommend you to upgrade your system in that case.

The next four gadgets control various functions of ViNCEd:

Chunky pixel graphics

Selects the type of graphics system you work with. By default, ViNCEd is optimized for the Amiga bitplane type of graphics, and it only scrolls the needed bitplanes of the window. But with most graphic cards, this will in fact slow down the operation, so check this gadget if the rendering is done by a graphics extension.

It is also possible to specify the type of graphics on the window open time by the "CHUNKY" and "PLANAR" open-path options . The first one selects "chunky graphics" like a checked gadget would, and the second switches back to the default, Amiga type graphics type.

REMARK: ViNCEd does not crash if the type does not match, it just gets slow. (Unless your graphics card software is not compatible to Commodore's programming guide lines)

Line break at right border

If a line is too big to fit in the visible part of the window, the additional part is usually kept hidden in the right border of the window. It can be seen by scrolling the window with the scroller gadget at the bottom of the window, or by moving the cursor with the cursor keys in the scrolled off part. Lines of output are only split, if they do not fit in a complete ViNCEd line, which is about 240 characters wide. However, some programs don't expect such a behaviour and want lines to break at the right border. But once lines have been broken, they are, well, broken, and can't be edited as a whole. If, for example, the window gets resized, the line doesn't get broken again and will stay in its original state. Especially, some parts of the line MAY get obscured if the window gets smaller. This is different from what CON: does with broken lines: They are still intact as a whole and get reformatted if the window gets resized.

This option should be only turned on for compatibility reasons, and is supposed to be "hack"; there is also a CSI sequence to enable or disable this temporary. It also influences "output", that means: Editing won't change.

To enable wide scroll (i.e. the usual ViNCEd wide display), use "CSI >? 19h" to disable it, that is allow line breaks at the window boundary, use "CSI >? 19l"

Inhibit horiz. scrolling by DOS output

If you activate this gadget, ViNCEd does no horizontal scrolling if the cursor moves past the right edge by some output command, instead, it just gets invisible. This is useful when long lines are printed, and flickering by horizontal scrolling is discouraged. On the other hand, it might be quite irritating to leave the user with the cursor scrolled off, and hence invisible.

Editing itself is not affected by this flag, it only controls what happens if the cursor gets past the right edge by some printing command.

Again, consider this as a "hack" for special programs.

There is also a CSI sequence to enable this feature on request. Use "CSI >?18l" to enable this hack, and "CSI >?18h" to disable it again.

VT-220 compatibility mode

ViNCEd has its onw interpretation of some control sequences, mostly oriented on the CON: interpretation, plus some extras that are "transparent" (i.e. not visible) by the standard set.

This is partly not compatible to the VT-220 way of interpretation, which is usually not used by the Amiga, but by most UNIX systems - A terminal program would be a good example for a program that relies on VT-220 features. A complete list of all control sequences understood by ViNCEd can be found here , together with remarks how their interpretation changes in different modes.

While I advice you NOT to turn on this by default, it might be quite useful from time to time.

To turn this mode on, use the CSI sequence "CSI >?2h", to disable it again and use the default CON: way of interpretation, use "CSI >?2l".

Again, this setting DOES not change the edit-behaviour of ViNCEd, it just concerns the output.

1.62 The System Page, part 2

Here you find the description of the second system page, the first one can be found here .

Destructive DEL and BS

Selects the function of the backspace (ASCII 08) and delete (ASCII 7F = 127) control sequence when printed. The function of the keyboard keys DOES NOT CHANGE.

By default, ViNCEd emulates the behaviour of the CON: window handler. Thus the backspace control code only moves the cursor backwards one position, and the DEL sequence is printed as a smear symbol, something like a checkered box. If you select this gadget, the backspace works like the keyboard backspace, i.e. it erases the character to the left of the cursor, and the delete control sequence does the same to the character under the cursor.

Most programs won't expect this when printing a backspace control sequence, so I advice you not to turn on this option. However, if explicitly needed, this option can be turned on by a CSI sequence . It is "CSI >?24h" to turn the destructive backspace and delete on, and "CSI >?24l" to turn it off again.

Insertion mode for DOS output

Selects if printed text should overwrite the window contents - the default - or should be inserted. If this gadget is checked, the insertion mode is used, although this is not compatible to the CON: behaviour, so use with care. As the flag above, this DOES NOT change the keyboard insertion or overwrite mode, which is controlled by a different flag, you'll find here .

Most software does not expect insertion mode, so leave this flag off if you don't have a good reason not to do. However, there is a control sequence - which is part of the VT-220 emulation - to turn this flag on: "CSI 4h". To switch back to overwrite mode, use "CSI 4l".

Notify DOS about paste

If you press Amiga-V in a ViNCEd window, the clipboard contents is pasted into the keyboard buffer. (This functionality is provided by the ConClip program for the CON: windows, which is not needed for ViNCEd.) However, a program might choose to do the insertion for itself. If you select this gadget, ViNCEd will not insert the clipboard contents but will instead send a Paste Report "CSI 0 SPC v" (SPC is the blank space, ASCII 20 or 32 decimal). It is then up to the program to do the right thing. However, regardless of this flag, Cut, Copy and all the remaining block operations will work as normal, but the middle mouse button is also affected, since it inserts the clipboard, too.

Since most programs can't handle a Paste Report, leave this flag off unless you have good reasons. However, a program might choose to disable temporary the automatic paste by ViNCEd. This is done by the CSI sequence "CSI >?25l". Use "CSI >?25h" to turn this on.

REMARKS: The paste behaviour of ViNCEd does NOT change if a window is in raw mode instead of cooked mode . This is contrary to what CON: windows do, they don't paste in this case, but send a Paste Request, as a plain console.device window would do.

There is a second flag related to the one presented above, which can't be set by default cause it is even more restrictive. If you send the CSI sequence "CSI >?27h" to ViNCEd, all the block operations, including Cut and Copy and the selection of blocks must be done by you. Special CSI sequences will be send to indicate a Cut or Copy request, and the block marking is totally your own business, i.e. you have to listen to the mouse movement and -keys. Send a "CSI >?27l" to restore the normal operation.

Disable full screen edit

Make ViNCEd into a line editor, somewhat like CON:, but even more restrictive. The user is not allowed to leave the current line, even by a RETURN (or better, a "Send"-keyboard command...). Since this usually trashes the shell output, it is recommended to leave this flag off.

However, a program might choose to turn on this flag temporary, if, for example, a file name in an editor must be entered and it is forbidden to return to the edited text until the input operation is completed. In this case, turn the line editing on by the CSI sequence "CSI >?3h". To use the full screen editor again, turn off this flag by "CSI >?3l"

TO SAY IT AGAIN, GUYS: THIS IS NOT FOR THE SHELL

The shell needs a fully working RETURN (or send) key. This mode is intended for programs that setup input masks on the screen, hence forcing the user in these masks without giving a chance to leave them. Even the RETURN key is not allowed to leave the mask, and hence the current line. THIS IS NOT WHAT YOU WANT FOR THE SHELL !

Use the XTerm cursor flag instead, it is here .

1.63 The System page, part 3

Here you find the description of the third system page of the SetVNC program. The topmost first page is here .

The third page is used to setup the system macros used by ViNCEd. They are somewhat similar to the user macros that can be setup by the macros pages , but cannot altered by the user thru the macros menu in a ViNCEd window.

To find out more about how macros (and the closely relied buttons) are expanded, and about the special patterns and control sequences that can be used in them, read here .

Quit shell

This macro is expanded by ViNCEd on a request to close the window if a shell is waiting for input in this window. This is only done if the "Call macro to close window" flag in the shell page is selected, otherwise a End Of File condition is send. Something like a "EndCLI" command should be placed here. (The code $02 which is present in the default setting of this macro is the Alt-DEL key on the keyboard and erases the complete line before inserting the macro itself.) A more sophisticated use of this macro would call a shell script which, for example, could erase temporary files created by a compiler session, and much more.

If you leave this macro empty, an EOF condition will be sent to the shell.

There's a second macro that is used to close a ViNCEd window, which is defined on the next page . It is invoked if the shell is executing a command instead of waiting for input.

New window

This macro gets expanded on selection of the "New Window" item in the project menu . It should bring up a new shell window, so something like a "NewCLI" command is the best choice. If you use a custom shell, enter here a similar command.

A second use of this macro would be to invoke a new editor window in a compiler session.

Get help

Invoked if the user presses the Help key or selects the "Help" item of the project menu . This macro should bring up some help related to the current work. By default, SetVNC is called to display the ViNCEd help pages, i.e. this guide. See also the command line arguments of SetVNC.

A good use of this macro would be to bring up the help pages of a compiler.

Edit settings

Gets expanded if the user selects the Settings item in the project menu. By default, this brings up the SetVNC preferences editor to change the settings of the window, but it might be quite useful to bring up a settings editor of a compiler.

Fork new shell

This one is unique cause it is not expanded into the keyboard buffer like all the other macros. Instead, it is executed by the shell.

The macro, which is part of the job control of ViNCEd is used when a new shell is requested by a press to Ctrl-Z. It should bring up a new shell in the current window. If you use a custom shell, insert here a command similar to "NewShell WINDOW=*".

REMARK: Since this macro is executed rather than inserted as keypresses, it should be terminated by a linefeed sequence "\n" (ASCII 0A = 10) instead of a RETURN key press "\r" (ASCII 0D = 13).

another REMARK: The execution of this macro causes usually a re-execution of the S:Shell-Startup sequence. A usual problem with this is, that you have to make this script re-entrant, i.e. stable for re-execution. One of the problems that may arise is the setting of pipe characters, namely the commands

Set _pchar | Set _mchar ||

Since on the second call, the definition of the pipe characters is interpreted as a pipe itself, this won't work. Change the lines to avoid the interpretation of the characters "|" and "||" by the shell:

Set _pchar "|" Set _mchar "||"

Note the double quotes!

1.64 The System Page, part 4

Here comes the fourth and last part of the big SetVNC system page. The first and uppermost page is here .

The fourth system page is mainly used for to control iconification of ViNCEd windows. It specifies both a default icon and a default icon name, but first one additional system macro:

Quit program

This system macro will be expanded if the user presses the close gadget of a ViNCEd window while a program is being executed in it. It can be used to abort the command by sending a Ctrl-C or to send an EOF condition by leaving it empty.

The last two macros define strings for the window iconification:

Icon Path

Enter a complete path to an icon (".info" - file) which should pop up if a ViNCEd window gets iconified. The position of the icon is also relevant, so it is probably a good idea to "UnSnapshot" it before using it with ViNCEd.

If this string is left black, ViNCEd will use its default icon, a simplified shell like window.

Icon title

This string specifies the default title used for the icon, i.e. the name that appears together with the icon on the workbench screen. If this string is left blank, the window title appears by default. However, a program might choose to setup its own title by an ESC sequence that overrides this setting.

Use "ESC] 1;title BEL" to setup a new icon title.

1.65 SetVNC command line arguments

SetVNC comes with a big bunch of command line arguments used to change settings "on the fly", i.e. without going thru the editor . You can always ask for the complete argument mask if you enter "?" as first and only argument to SetVNC:

SetVNC ?

There the complete list of arguments:

QUIET

Suppress verbose output, print only error messages. DO NOT REDIRECT SETVNC OUTPUT TO NIL:, SINCE THE OUTPUT STREAM IS NEEDED TO FIND THE CONTROLLING VINCED WINDOW!

MOUNT

Meta argument that comes with with two subarguments. It is used to replace a (mounted) handler by ViNCEd, usually CON:. A line like

SetVNC mount override as CON

should be present in your startup-sequence if you want to do so.

MOUNT AS

The "MOUNT" argument takes at least one sub argument "AS" which specifies the name of the handler that should replaced by ViNCEd. The keyword "AS" itself is optional.

MOUNT OVERRIDE AS

Tells SetVNC to replace the specified handler, even if already mounted, like the CON: handler of the system. Thus, "OVERRIDE" is almost always needed.

HELP

Ask SetVNC to pop up the help database. This works only if:

1) The path of the help database is specified correctly in the system page of SetVNC.

2) The default tool of the help database icon is set to the complete path of your favorite browser tool. This should be "SYS:Utilities/MultiV for Workbench 3.0 and up or "SYS:Utilities/AmigaGuide" for all others. To change this, select the database icon on the workbench, press Amiga-I to get information about the icon and enter the path of the brower into the "Default Tool" string gadget. Save the changes.

BACKGROUND

Part of the job control command set of SetVNC. Used to bring the uppermost shell to background.

FOREGROUND

This meta argument is also part of the job control and is used to bring a shell to foreground. It has to sub arguments, one optional and one required:

FOREGROUND

Takes one argument, the CLI number of the shell that should be brought to foreground. CLI numbers can be listed with the "Status" command.

FOREGROUND OTHER

Works like the arguments listed above, but can also bring processes in different windows to foreground. This should be used with care!

FREEPOINTER

Some programs arbitrate a pointer to the underlying intuition window from ViNCEd, making it impossible for ViNCEd to close the window. This option, as a hack, gives back the window pointer BY FORCE. You should only do this if YOU REALLY WANT, and you really know what you're doing - otherwise it is likely that some other programs run from ViNCEd will crash. The typical application for this command line argument is as a workaround for programs, that do not give the intuition window pointer back, e.g. CBM's "more".

SETCONSOLE

This option sets the processes controlling console task to the one supplied as output to SetVNC. This could be useful if a process opens a console using the "*" symbol, which redirects the output then to this specific console, and hence to a given owner, to setup a foreground/background link for it.

The terminus "owner" is a part of ViNCEd's job control mechanism, read about it here , or here .

LOAD

A meta argument which specifies which preferences to load into the preferences editor . By default, the current window's settings are loaded. This argument can also be used to load previously saved preferences into the current window, if the editor is not invoked by the "MODIFY" command.

LOAD PREFS

Load the currently used global preferences. These are not necessarily the windows settings. Works somehow like the "Last Used" button in the SetVNC editor .

Do not mix this sub argument with the top-level argument "PREFS", which does something different.

LOAD GLOBAL

Does the same as the "PREFS" sub-argument and load the global prefs.

LOAD WINDOW

Load the window's settings into the editor. This is also the default, if the top level argument "PREFS" (not the sub argument listed above) is present.

LOAD DEFAULT

Load the factory defaults into the editor or the window. Works like the "Default" button in the editor.

LOAD LAST

Load the last saved settings into the editor or the window. Works like the similar named button in the editor .

LOAD FROM

Load the settings from a given preferences file, which must be specified beyond both arguments. This is most useful to load session-specific defaults into the window. I.e.

SetVNC load from compiler_prefs

loads the window's settings from a previously saved file named "compiler_prefs".

PREFS

Specify that SetVNC should be run as a preferences editor , not as a window settings editor. This does not make a big difference, but chooses for example which prefs are load into the editor by default. If you do NOT specify "PREFS", the window's settings are edited, if you DO, the global settings are load by default.

See also here and here .

Two command follow that are used to save and restore parts of the ViNCEd buffers to files:

PUT

This command can be used to save parts of the ViNCEd buffers to files. Two sub arguments are valid here:

PUT SCREEN

Save the contents of the window buffer as ASCII text to a given file. The text rendering of the display buffer is embedded in the ASCII output as CSI sequences. As an example: The command line

SetVNC PUT SCREEN ram:buffer

will save the buffer to the file "ram:buffer".

PUT HISTORY

Save the contents of the history as ASCII text to a file. This command is used by the ViNCEd history script in the S: drawer.

The command

SetVNC PUT HISTORY *

will, for example, print the history to the ViNCEd window itself.

GET

This command option is the opposite to the "PUT" argument. It will load an ASCII file to one of the ViNCEd buffers .

GET SCREEN

will load a given file to the display. Only very few CSI sequences are parsed by SetVNC, to speedup the loading. Namely, CSI m (set rendering) is the only interpreted sequence here and selects the colors of the text in the display. All other sequences are directly inserted into the window.

GET HISTORY

will load an ASCII text file as the history of the ViNCEd window. The old history will be lost.

RESET

Resets the current window by the "ESC c" control sequence , but does not change window's settings itself.

The next meta arguments are similar to most check gadgets you find in the prefs editor . They all take a boolean sub argument which can be either "ON" or "OFF". Their full functionality is not described here, but instead in the pages of the editor.

DOSCURSOR

Selects the "Dos Cursor Mode" also found in the menu or in the edit page under the name "Unrestricted cursor movement".

CRINSERT

Also named "CR inserts below at start of line" in the edit page .

TABMOVE

Enables or disables the movement by TAB, also found in the editor under "TAB moves, no insertion".

OVERWRITE

Select "Overwrite Mode" like the menu item , or like the "Overwrite text" setting in the editor .

WRAP

Enable the "Wrap Around Buffer" of the menu and the "History buffer wraps around" of the editor .

SMARTCLOSE

Selects the window closing by macro invocation, as the menu item "Smart Close" or the editor setting "Call macro to close window".

SAFERCLOSE

Selects the "Safer Close" menu item which is similar to the "prevent accidental window closing" in the editor .

DISABLEPROPX

Prevents ViNCEd from adding the horizontal scroller by default to each window. THIS FLAG WILL NOT CHANGE THE CURRENT WINDOW LAYOUT, but will change the defaults. Read on here for more on that.

DISABLEPROPY

Same as the above, except that it disables the vertical scroller by default. More can be found at the same location .

CLOSEREQ

Enables the "Close requester" as in the third shell page . Useless unless you enable the "Safer Close" as well.

CLOSEQUEUE

Enables or disables the tricky "Auto Close Queue", similar to "Don't send EOF until everybody waits" in the editor .

CUTUSER

Switches the "Cut Inputs Only" in the menu , or "Don't write printed text into clipboard" in the editor .

REBUILDDELAY

Enables the delayed window refresh, like the menu item "Rebuild delay". This flag can be found on the timing page , too. It is called "Allow delayed window refresh" there.

AUTOCOPY

Selects the automatic text copy of marked blocks. Can be found in the menu as "Auto Copy" or in the editor as "Implicit copy after text marking".

ALTMOVE

Changes the way how the cursor keys work. They control either the history or the window buffer . This flag is similar to the "Alternate Movement" flag in the menu and to the "Swap ALT-cursor and cursor keys" in the editor .

ALTDEL

Maps the delete functions "Erase start of line" and "Erase end of line" to the keyboard combinations "Alt+BS" and "Alt+Del". The former functions "Erase word" and "Erase user input" can be reached with "Shift+Alt+BS" and "Shift+Alt+Del". See also here for the corresponding flag in the editor.

NUMPADMODE

Selects the numeric key pad for cursor control if turned on. The keyboard layout in this mode is here. . Can be turned on as well as with the prefs editor as with the CSI sequence "CSI ?1 h".

SCROLLBORDERS

If turned on, cursor movements will be truncated to the display buffer, i.e. the cursor won't move into unused lines. This flag is also available in the third edit page , as "Don't scroll into the border".

TYPEAHEAD

Will enable the type ahead buffer. Typed characters won't appear on the display immediately, but will be delayed until a program is requesting for user input. Also available as "Invisible type ahead" in the prefs editor .

TABEXPAND

Enables or disables the file name expansion by the TAB key (more precisely: Selects whether TAB or Ctrl+TAB is the "Expand"-key). Similar to the menu point "Expand with TAB" and the editor setting "TAB filename expansion".

DOUBLETAB

Enables or disables the "Double TAB Requester" setting, found in the menu . The same is done by editor setting of the same name.

AMBIGREQ

This is the flag for the "Request if not Unique" setting in the menu , which is also controlled by SetVNC on this page. Called "Requester if expansion is ambiguous" there.

VNCLIST

Same as "List ViNCEd Matches" in the menu , used to show ViNCEds internal list of matches instead of the directory. Can be found here in SetVNC, named "Show ViNCEd matches, not directories".

FULLEXPAND

Enables or disables the full expansion of the file name pattern by the first press to TAB (or the "Expand" key). Named "Expand Fully" in the menu , and "First TAB expands fully" in the editor .

DIRSFIRST

Changes the ordering of the ViNCEd TAB expansion list (sorts in directories prior to devices). See the "Show matching directories first" flag of this editor page.

NODEVS

Do not insert devices into the ViNCEd TAB expansion list, unless the user asks explicitly for them by giving a pattern ending with a colon. This flag is present as "Devices don't match implicitly" in the third shell page of SetVNC.

NOINFO

Do not add icon files (".info" files) to the ViNCEd completion list unless explicitly asked for them with a pattern ending with ".info". Also present in the fourth shell page .

PARTIALPATH

Same as the similar item in the settings menu , prevents ViNCEd from searching the complete path for expansions of commands. Can be set from within GUI as well, see here .

TRUNCATEPATH

If set to "on", truncates the search path for TAB expansion of commands even more, depending on the "PARTIALPATH" flag above. If this one is set and "PARTIALPATH" is "OFF", the current dir, C: and the command resident list is used for searching. If both flags are enabled, ViNCEd does not distinguish between commands and other arguments and looks only into the current directory.

LINEBREAK

Enables or disables swapping of the "Line Break" (ALT+RETURN) and the "SEND" (RETURN) key . This might be useful for editors and emulates somehow the MPW shell keyboard layout of the Apple computers: The "SEND" key does no line breaking, but instead sends the line contents to the shell - this is what usually happens if you press RETURN. However, in an editor RETURN does not send data, but usually splits the line at the cursor position. This function is assigned to Alt+RETURN by ViNCEd. If you want to use ViNCEd as an editor, it is probably better to use the keys the other way round - this is done by this setting.

Because it's so ugly, it can't be enabled by default, i.e. there is no editor flag for it. However, there is a CSI sequence to enable or disable this by hand. Use "CSI >?28h" to enable this, or "CSI >?28l" to return to the default setting.

FORBIDICONIFY

If turned "ON", it does forbid the installation of the iconify gadgets in ViNCEd windows. It DOES NOT change the setting of the current window, but only changes the preferences; thus you need a SAVE argument to make some sense out of this.

This argument is similar to the "Don't add iconify gadget by default" setting of the editor .

CHUNKYPIXEL

Controls the type of graphics output used by the window. Usually optimized for Amiga bitplanes, ViNCEd uses graphics routines optimized for "chunky" graphic systems, such as graphic cards, if this flag is turned "ON".

Similar to the "Chunky pixel graphics" flag in the system page of SetVNC.

SHELLMODE

Setup the current window as shell window, and select the shell mode as default, if the flag is turned "ON".

Similar to the "Use shell mode by default" flag in the editor , but does also change the current window mode, not only its default.

NODEFAULTCLOSE

Selects whether a close gadget should be added to windows by default or not. DOES NOT change the layout of the active window, so without "SAVE"-ing the changed settings this leads to nothing.

This flag is named "Don't add close gadget by default" in the editor .

NOPRINTSCROLL

Giving the sub argument "ON", disables horizontal scrolling caused by printing. Similar to the "Inhibit horiz. scrolling by DOS output" in the editor or the CSI sequences "CSI >?18l" or "CSI >?18h".

SMALLDISPLAY

Enables word wrapping at the right edge of the window instead of the right edge of lines, which are approx. 240 characters wide. This flag is also available in the editor , called "Line break at right border", or as the CSI sequences "CSI >?19l" (to turn it on) or "CSI >?19h".

DOSERASE

Enable the destructive backspace and delete control sequences by the boolean sub argument "ON". Same goes for the editor setting "Destructive DEL and BS"; the identical function is provided by the CSI sequences "CSI >?24h" or "CSI >?24l"

DOSINSERT

Toggles the insertion mode for printed text like the "Insertion mode for DOS output" gadget in the editor . Also available as CSI sequences "CSI 4h" or "CSI 4l".

AUTOPASTE

Used to enable or disable the immediate copying of a block into the clipboard after marking it. This setting can also changed by the settings menu item "Auto Copy" and by the editor gadget "Implicit copy after text marking".

DISABLESCROLL

If turned "ON", disables the scrolling at the bottom of the window. Similar to the CSI sequences "CSI >1l" and "CSI >1h".

DISABLEWRAP

Disables word wrapping at the right edge of the line or window (dependent on the setting of SMALLDISPLAY, see above). If turned on, word wrapping is disabled, and a huge line will never be split into two lines.

Word wrapping is also controlled by the CSI sequences "CSI ?7l" and "CSI ?7h".

XTERMCURSOR

If turned on, this selects the XTerm like cursor movement restrictions, which are also setable from the SetVNC program, here .

This option can be controlled by a CSI sequence as well, it is "CSI >?30h" to enable and "CSI >?30l" to disable it.

FREECLICK

If this flag is on, setting the cursor with a mouse click does not block the output, regardless of whether the XTerm mode is enabled or not.

This flag is, of course, also available in the prefs editor, see the forth edit page .

SCROLLTOBOTTOM

Keeps the bottom line always adjusted to the bottom border of the window, even though if the window gets resized. This flag is also available in the editor, see here .

RAWSCROLLERS

Enables the scrollers, even in the RAW mode. This allows you to scroll back to older parts of the window buffer in the RAW mode. However, you won't be able to move the cursor since it is under control of the program running in this shell. This flag works like an implicit XTerm mode for RAW windows. Read also the description of the forth edit page .

VTMODE

Selects VT-220 compatibility mode if turned "ON". The default is the Commodore CON: emulator. There is both a flag in the editor, called "VT-220 compatibility mode" and the CSI sequence to control this setting

ANSIMODE

Enables the ANSI coloring scheme . In this mode, the color codes of the CSI m set render control sequence are interpreted as ANSI color indices, as defined by the ANSI standard. The default layout is then white text on black background.

This mode can be activated by CSI >?8h as well, and is also selectable by the open path argument ANSI .

ANSIREVERSE

Enables reverse video for the ANSI mode, i.e. swaps the ANSI colors 0 (black) and 7 (white) to obtain black text on white background. This flag does nothing if the ANSI MODE itself is disabled!

Read also here for the editor page of this flag.

ROWLOCK

If enabled, then the user is not allowed to leave the current line. This line edit mode is even more restrictive than the CON: line editor, so use with care! This flag is equivalent to the editor flag "Disable full screen edit" and the CSI sequences "CSI >?3h" and "CSI >?3l".

UNDERLINE

Switch the cursor to the underscore block if turned "ON". Also controlable by the "Underline cursor" in the editor or by the CSI sequences "CSI >?4h" and "CSI >?4l".

BLINKING

Turn on the cursor blinking. Same is done by the "Blinking cursor" gadget in the editor , or by the CSI sequences "CSI >?6h" and "CSI >?6l".

DISABLEMMB

Disables the middle mouse button and makes it free for other applications. Same as this GUI setting. Read also here about the replacement key combination.

NOBSSTART

Disables the backspace key at the start of a line: No line merging will take place. Same as the CSI sequence "CSI >?20h". This flag is also in the GUI .

The next three settings control the sizes of the ViNCEd buffers . The same function is also provided by the settings menu , and by the Buffers page of the SetVNC editor. The buffer sizes are given in lines as a numeric sub-argument.

BUFFERSIZE

Set the size of the history in lines, valid arguments range from 8 to 1024.

LOWERSIZE

Set the size of the lower display buffer which keeps lines in the and below the window.

UPPERSIZE

Set the size of the upper display buffer . It keeps lines scrolled over the top edge of the window.

The next arguments are used to setup a default monitor for ViNCEd custom screens . This can be done by the GUI as well:

MONITORID

Set the default monitor by monitor ID. This works always, even for workbench 1.3 (but only the lower 16 bits are used in this case). Hex values are allowed here if preceded by "0x" or "$".

MONITOR

Set the monitor by name. This name is part of the localization, and hence depends on the active language. Use the system screen mode requester to find out the names of the monitors.

The four arguments that follow are supposed to control the ViNCEd timing, like the Timing Page of SetVNC. More about their meanings can be found there . The delay intervals are given in milliseconds (ms).

DELAYINTERVAL

Set the internal ViNCEd delay used for a workaround of an intuition bug. The value is only used with workbench 1.3 and below. Named "Intuition delay" at the Timing page .

REBUILDTIME

Used if REBUILDDELAY is turned on. Then this is the time between the start of the delayed refresh to the refresh itself. This setting is also present under the name "Rebuild delay" at the Timing page .

SCROLLTIME

This is the minimal scrolling speed needed to avoid delayed refresh. Named "Vertical scroll threshold" at the timing page , and only needed if REBUILDDELAY is turned ON.

BLINKSPEED

Sets the blink interval of the cursor, if used. Found under "Cursor blink speed" in the Timing page .

Now two arguments to setup the qualifiers for the icon drop feature. They can be changed more user friendly thru this prefs editor page:

PATHONLYQU

Specify the qualifier used to insert only the path of a dropped icon. The argument is supposed to be a number, add the following values:

Shift 1

Control 8

Alt 16

Commodore 64

Again, hex numbers are valid here.

NAMEONLYQU

Setup the qualifier for inserting the name of a dropped icon. The numeric value used here is calculated again by adding some of the numbers of the table above.

And now some miscellaneous arguments:

MODIFY

This argument invokes the graphical settings editor of SetVNC. It MUST be given if you want the editor, unlike to previous versions of SetVNC. Only for backwards compatibility, the editor is also invoked if no argument is given at all.

SAVE

Another meta argument like "LOAD". It saves the edited preferences back to a location of your choice.

NOICONS

Optional sub argument of SAVE, inhibits the creation of an icon.

PREFS

Saves as global preferences to the default place on disk. Works like the "SAVE" button in the editor .

GLOBAL

Uses the preferences globally, but does not save them to disk. Works like the "USE" button in the editor . This is the default if you gave the "PREFS" argument to invoke SetVNC as the preference editor .

WINDOW

Installes the changes only into the window. This is the default if you did not use the "PREFS" argument, hence invoked SetVNC to modify the window settings .

TO

Takes a file name as another argument and saves the preferences to that file.

The next meta argument "TABHOOK" is only here for backwards compatibility and controls the TAB expansion . More about this argument can be found here .

TABHOOK INSTALL

Selects shell mode for the current window and enables the TAB expansion .

TABHOOK REMOVE

Disables TAB expansion , but does not cancel the shell mode.

TABHOOK QUIT

Same as above, but also disables the shell mode and prints a warning message that the TAB expansion is now build-in, and can’t be removed totally.

1.66 Job control with SetVNC

What is this, "JOB Control?"

Well, it is possible to run more than one shell in a SetVNC window at a time. This was also "sort of possible" with the old CON: handler, but usually two problems arises:

-Which program receives the input you type?

-If you run more than one shell in the window, how to avoid that both outputs trash the window?

ViNCEd, and UNIX, solve the problems by defining only one shell (or better "process", even better: Owner) as the "Foreground" process, which gets all the input, and everything else as "Background".

Background processes don't receive input, and they are not allowed to print to the window. If they try to print to the window, they get "suspended", i.e. send to wait state until you send them back to foreground.

This has nothing to do with the concept of "Background" introduced with the RUN command. Commands started this way can't be distinguished by ViNCEd from the shell they are forked off.

Instead, the identification works by "named consoles": Don't use a path like "*" or "CONSOLE:" for a new shell, but give them a unique name, like "CONSOLE:myproc". This is fine for OS 2.04 and above, but does not work for 1.3 and below... well, UPGRADE in this case.

The old "*" stream identifies always the foreground shell, the "CONSOLE:" always the stream that ViNCEd was started with, and, finally, named consoles, well, identify themselves, i.e. all processes using "CONSOLE:myproc" are always in common in "foreground" or "background".

To fork a new shell in the same window, use something like

Run NewCLI Window=CONSOLE:myshell

where myshell is some unique name.

Why do I need the "RUN" command?

The reason is a bit technical... Suppose you omit the "RUN". Then a new named console (slang: " Owner ") of the name "myshell" is created, and the new shell process prints its startup message into this stream - well, it tries to. The point is, that opening a new stream DOES NOT send the opening process to foreground, and hence the new shell is suspended. Silly enough, cause the new shell is halted, the old can't continue execution, cause it waits for the "NewCLI" command to return. You end in a so called "deadlock" situation! To break this deadlock, open a new shell window, type "status" and find out the command number of the new shell, maybe make a guess. Then send this new shell to foreground, using the "SetVNC other foreground" command described below. The new shell is then able to print and the deadlock is gone.

However, the best choice is to use the "fork" script in first place - it does all this for you.

Well, now - how to put a shell process in foreground or background:

The SetVNC program has two arguments for that:

SetVNC Background

Puts the current shell into background. However, this will only work if there is something to put into foreground instead and may fail for that reason. In fact, it does not make much sense to put the only program running in a window into background, since then you are lost with a useless window.

The same is done by a small script named "bg", for UNIX freaks. Well, actually "bg" does not EXACTLY the same like the UNIX "bg", but something similar...

SetVNC Foreground

Takes one argument, the CLI number of the process to put in foreground. This process must belong to the actual window, i.e. you can't put a process into foreground that does even use the window you type in (logically). As a second effect, since only one program can be in foreground, the shell you type in is send to background.

By the way: The CLI numbers are printed by the "status" command, which is aliased by the "real UNIX freak" to "ps".

The same is done by the script "fg", which takes the same argument. Again, this is not EXACTLY what UNIX "fg" does, but nearly.

SetVNC other Foreground

This is a more powerful combination than the above, since it can also bring up shells to the foreground that have been started in a different ViNCEd window. The operation needed for that is a bit tricky, hence you should use this with care. It is also no "good style" to type a command in one window to control a second. The only reason when one should use the "other" option is to break a deadlock situation, as described above.

More about job control and the special key Ctrl-Z can be found here.

A note for authors about named consoles is here .

1.67 Tooltypes of the SetVNC program

The SetVNC program can also invoked from the workbench, mostly used as a preference editor . It recognizes the following tooltypes, which can be set in the usual way, by "Information" in the workbench menu:

WINDOW=path

Specify the window SetVNC should appear in. This tooltype MUST BE PRESENT, and it MUST be a path to a ViNCEd window.

Starting with the 40.12 release of SetVNC, the guide path is also kept in an environment variable. You shouldn't modify it by hand, so its location is not given here.

ACTION=args

Gives commands what SetVNC shall do. The "args" string is a list of shell arguments. They are passed to SetVNC as if called from shell. A complete list of all arguments is here .

Most useful is the argument list "PREFS MODIFY", which invokes SetVNC as a preferences editor . Without them, SetVNC would just change the settings of the window it was created in - which is rather senseless.

HELPPATH=path

Specifies the complete path to this guide. In fact, this is the place where it is kept, aside from the usual preferences. The guide is used when you press a "Help" button in the editor , or you ask SetVNC to display the guide with the "Help" shell argument .

1.68 The Index